THESIS

USMC INVENTORY CONTROL USING OPTIMIZATION MODELING AND DISCRETE EVENT SIMULATION

by

Timothy A. Curling

September 2016

Thesis Advisor:	Arnold Buss
Thesis Co-Advisor:	Javier Salmeron

This thesis was performed at the MOVES Institute

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704–0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2016	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE USMC INVENTORY CONTROL USING OPTIMIZATION MODELING AND DISCRETE EVENT SIMULATION	**5. FUNDING NUMBERS**
6. AUTHOR(S) Timothy A. Curling	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Marine Corps Installation and Logistics Command is seeking assistance to improve operations within Marine Corps Maintenance Production plants. The problem addressed in this thesis deals with production lines: there must be a proper balance of parts on hand and inventory costs to ensure optimal production output. This problem becomes increasingly difficult to solve as production-line complexity increases and overall budget flexibility decreases. As the Marine Corps enters a time of fiscal austerity and reduced overseas combat operations, it is critical to optimize its processes so major end items are refurbished in the quickest and most cost-effective manner, thereby ensuring maximum combat effectiveness.

This research focuses on developing a proof of concept analytical tool to better facilitate order management of repair parts. This tool integrates optimization and discrete-event simulation. This construct can potentially provide an effective means in improving order management decisions. However, the effectiveness of the tool is contingent on accurate vehicle condition history, parts order history, and/or future estimated parts shipping dates. Information derived from the analysis can be used to make recommendations for reorder policy, enable future model development, and improve the overall maintenance production process.

14. SUBJECT TERMS Optimization, Discrete Event Simulation, inventory management, Marine Depot Maintenance Command	**15. NUMBER OF PAGES** 99
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01-280-5500 Standard Form 298 (Rev. 2–89) Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

USMC INVENTORY CONTROL USING OPTIMIZATION MODELING AND DISCRETE EVENT SIMULATION

Timothy A. Curling
Major, United States Marine Corps
B.S., University of Utah, 2006

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN
MODELING, VIRTUAL ENVIRONMENTS, AND SIMULATION**

from the

**NAVAL POSTGRADUATE SCHOOL
September 2016**

Author:	Timothy A. Curling
Approved by:	Arnold Buss Thesis Co-Advisor
	Javier Salmeron Co-Advisor
	Peter J. Denning Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Marine Corps Installation and Logistics Command is seeking assistance to improve operations within Marine Corps Maintenance Production plants. The problem addressed in this thesis deals with production lines—there must be a proper balance of parts on hand and inventory costs to ensure optimal production output. This problem becomes increasingly difficult to solve as production-line complexity increases and overall budget flexibility decreases. As the Marine Corps enters a time of fiscal austerity and reduced overseas combat operations, it is critical to optimize its processes so major end items are refurbished in the quickest and most cost-effective manner, thereby ensuring maximum combat effectiveness.

This research focuses on developing a proof of concept analytical tool to better facilitate order management of repair parts. This tool integrates optimization and discrete-event simulation. This construct can potentially provide an effective means in improving order management decisions. However, the effectiveness of the tool is contingent on accurate vehicle condition history, parts order history, and/or future estimated parts shipping dates. Information derived from the analysis can be used to make recommendations for reorder policy, enable future model development, and improve the overall maintenance production process.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AAV	Amphibious Assault Vehicle
CPIOM	Critical Part Inventory Optimization Model
DES	Discrete Event Simulation
GAMS	General Algebraic Modeling System
IDE	Integrated Development Environment
MCLC	Marine Corps Logistics Command
MDMC	Marine Depot Maintenance Command
MOVES	Modeling, Virtual Environments, and Simulation
NIIN	National Item Identification Number
OMT	Order Management Tool
PEI	Principal End Item
POI	Poor Obfuscation Implementation
SO	Simulation Optimization

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my advisors Arnold Buss and Javier Salmeron. Without their dedicated and patient support, this thesis would not be possible. I also would like to thank Huntley Bodden and the staff of Marine Depot Maintenance Command who assisted me. I especially want to thank my wife, Joanna Curling, for her dedicated and loving support throughout my career. I would not be the person I am without her.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

This thesis develops the framework for an order management tool utilizing Discrete Event Simulation (DES) and optimization modeling. The goal is to improve order management policy within Marine Depot Maintenance Command (MDMC). This chapter provides an introduction of the overall project from which this thesis is derived and a background of the MDMC organization. This introduction yields an understanding of the existing problem being addressed. In addition, a brief discussion of the current techniques in which optimization and simulation are used to improve supply chain and inventory management processes is provided. This includes a discussion about previous research aimed specifically at improving overall MDMC plant operations.

A. PROJECT DESCRIPTION AND BACKGROUND

This section provides details concerning the origins of this thesis project, as discussed in the original project report [1]. The section discusses MDMC history, mission, and organizational structure.

1. Plant Utilization MCLC Project Background

This thesis grew out of an overarching project titled "Plant Utilization at Marine Corps Logistics Command" [1], sponsored by the Chief of Naval Operations with a program execution date of November 2013 through December 2014. The official customer of this project was Marine Corps Logistics Command (MCLC) Installations and Logistics. MCLC contacted the Naval Postgraduate School in March of 2013 soliciting proposals for research on improving plant capacity. The original proposal submitted by Naval Postgraduate School principal investigators included developing "mathematical models to guide plant design and utilization at MCLC, including optimal levels of physical capacity, equipment, manning and operations" [1]. After meeting with MDMC personnel at maintenance production plant Barstow in April 2014, the principal investigators determined the primary area of interest to be the management of critical repair parts. The difficulties of MDMC to procure critical repair parts results in stock-out situations. In addition, budgets are diminished because of unnecessary safety stock levels.

The primary focus of the project was to improve order management policy using analytical tools. This thesis focuses on the ordering of varying part types or National Item Identification Numbers (NIINs) using optimization. The goal is to minimize stock-out situations when stock levels are subject to budgetary constraints. As a result, maximum production output is facilitated. To this end, the optimization modeling tool named Critical Part Inventory Optimization Model (CPIOM) has been developed by the principal investigators to address this problem. CPIOM is discussed in more detail within Chapter II and Appendix A of this thesis.

The principal investigators solicited for graduate student participation to expand CPIOM utilizing simulation. This thesis focuses on using DES to provide additional insight into the problem, thereby allowing further improvement in inventory management policy at MDMC. As with the CPIOM, a detailed explanation of DES along with how it is merged with CPIOM will be provided in Chapters II and III of this thesis.

2. MDMC Background

MDMC is a subordinate organization of MCLC consisting of two maintenance production plants located in Barstow, CA, and Albany, GA. Until 2012, the two production plants operated independently. With a shrinking Department of Defense budget and demand by the Commandant of the Marine Corps to be good stewards of national resources, there is a greater need for MDMC to more efficiently and effectively reconstitute the Marine Corps with refurbished critical equipment necessary to accomplishing the Marines overall mission [2].

B. THESIS OBJECTIVE AND SCOPE

This thesis seeks to assist MDMC in improving its capability to maximize production mission by utilizing mathematical and analytical methods such as optimization modeling and DES. These methods have the potential to empower MDMC leaders in make better informed decisions concerning plant operations. The specific objective of this thesis is to implement the joint CPIOM and DES construct in order to facilitate order management decisions. In particular, the objective of CPIOM is to minimize the chance of parts being out of stock (i.e., reducing stock-out situations). The importance of this joint construct is explained in Chapter II, while Chapter III provides specifics of how the CPIOM and DES concept is implemented.

Additional objectives focus on more easily enabling MDMC to utilize the resources produced by this thesis. This includes facilitating data analysis by representing output data graphically, utilizing Excel spreadsheets for user input, and maximizing the use of open-source resources. During a visit with MDMC in December of 2014, it was noticed that Excel spreadsheets are used extensively by MDMC personnel. For this reason, they were also used for DES data entry to facilitate integration with order management operations. This is accomplished using the Apache open-source library POI [3]. In order to facilitate quick data analysis, the Order Management Tool (OMT) provides a graphical interface in which statistical output is represented through a series of histograms. These are generated using the JFreeChart open-source library [4]. Finally, the DES is developed utilizing the open-source library SimKit [5]. By using open-source resources, the dependency on proprietary software is reduced, thereby allowing MDMC more control in developing future iterations of the tool.

1. Thesis Scope

The overall scope of this thesis is to develop and implement the basic programming infrastructure required to produce an OMT. While the ultimate goal would be to facilitate all higher fidelity production lines for each Principle End Item (PEI) type and its associated NIINs, the initial scope in developing OMT must be limited to only a handful of repair part types or NIINs and a single PEI. This will allow for easier verification of CPIOM, DES, and joint construct functionality.

In order to establish a baseline in developing the CPIOM, the model is initially developed using historical data from five NIINs associated with the Amphibious Assault Vehicle (AAV). As discussed later in Chapter II, this small test sample provides critical information in relation to how the CPIOM formulates a distribution of required quantities for each NIIN. Both the CPIOM and DES in their current form are capable of handling a large number of NIINs and PEI platforms, the only limitation being one of computing resources. This is possible because the input variables of the CPIOM are the same as those of the DES. In addition, the current simplistic design of the DES does not require additional production processes to be modeled. However, this is assuming all associated data for each NIIN and PEI are available (i.e., the number of each NIIN for each PEI, price of each NIIN, initial quantity of each NIIN, etc.).

Enhancing the fidelity of the production process being modeled within the DES would require modeling multiple production lines, since each PEI platform would require a separate production line. That level of detail is beyond the scope of this thesis, which focuses on the most critical actions of disassembling and reassembling a PEI.

The core mission of MDMC is to produce PEIs in accordance with a designated output schedule. In meetings with plant managers located in Barstow, California and communication with MCLC leadership, it was determined that MDMC seeks to maximize the demand required of the output schedule by reducing stock-out conditions. There are many reasons why stock-outs may exist, which are discussed later within Chapter II. For this reason, CPIOM is specifically designed to minimize stock-out situations. The overall goal is maximizing production output.

2. Literature Review

The use of optimization modeling in conjunction with DES is a concept having several instances of literature studying its various implementations [6]-[11]. This concept is referred to as Simulation Optimization (SO) and is applied in a wide variety of industries, especially within the supply and logistics domain. In 2001, thesis research conducted at Naval Postgraduate School used SO to improve Marine Corps combat service and support element operations [7]. That thesis used optimization to determine the best use of resources to deliver supplies in a constrained time space environment. The results of the optimization are used as input variables within a DES. However, the idea of using an SO technique in a dynamic combat environment is most likely impractical, which is not the case in established supply chain networks.

In the area of supply chain network analysis, several studies exploring SO techniques are available. In 2006, researchers from the University of Vienna’s School of Business developed an SO framework in support of supply chain networks [6]. From Figure 1, one can discern that the idea in this research is to embed a simple optimization model within the framework of a complex DES. The optimization model improves the DES overall performance by adapting decision rules. After a few iterations, researchers found that they gained convergence to good-quality solutions within much less computational time than traditional optimization approaches [6]. Researchers from the University of St. Thomas created a four-step methodology for SO development of supply chain networks [8]. Figure 2 provides the basic outline of the methodology. This is essentially the same approach taken when developing the OMT (i.e., CPIOM developed first, DES developed second, integration of CPIOM and the DES third, and testing fourth). In 2008, research conducted at Arizona State University specifically focused on how to integrate optimization and DES models [9]. This research was not used in developing the OMT. However, it does provide insight as to how more complicated model integration can be achieved. For example, within a specific DES run there may be a requirement to run multiple optimization models. Design of the SO integration may not be a trivial task.

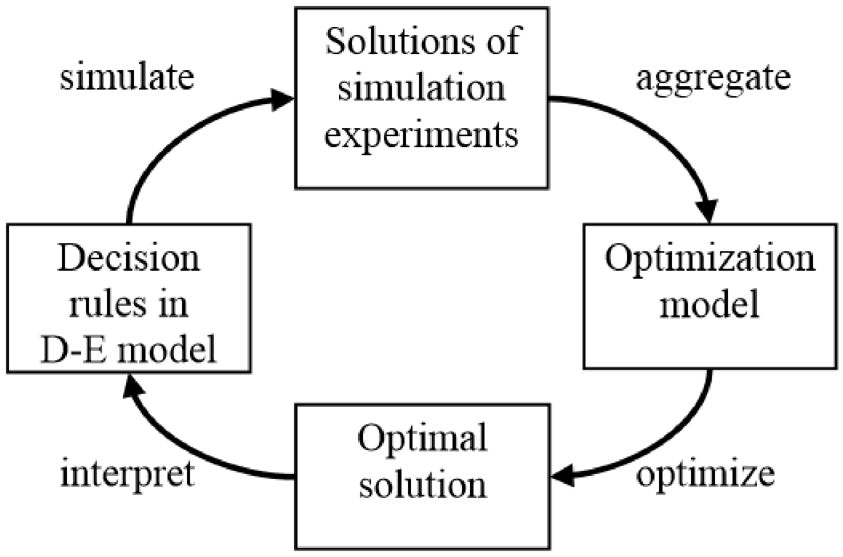


Figure 1. Interaction between DES and optimization model. D-E is the same as DES.

Source: [6] C. Almeder, M. Preusser and R. F. Hatl, “Simlulation and Optimization of Supply Chains: Alternative or Complementary Approaches?,” *OR Spectrum*, vol. 31, no. 1, pp. 95–119, 2009.

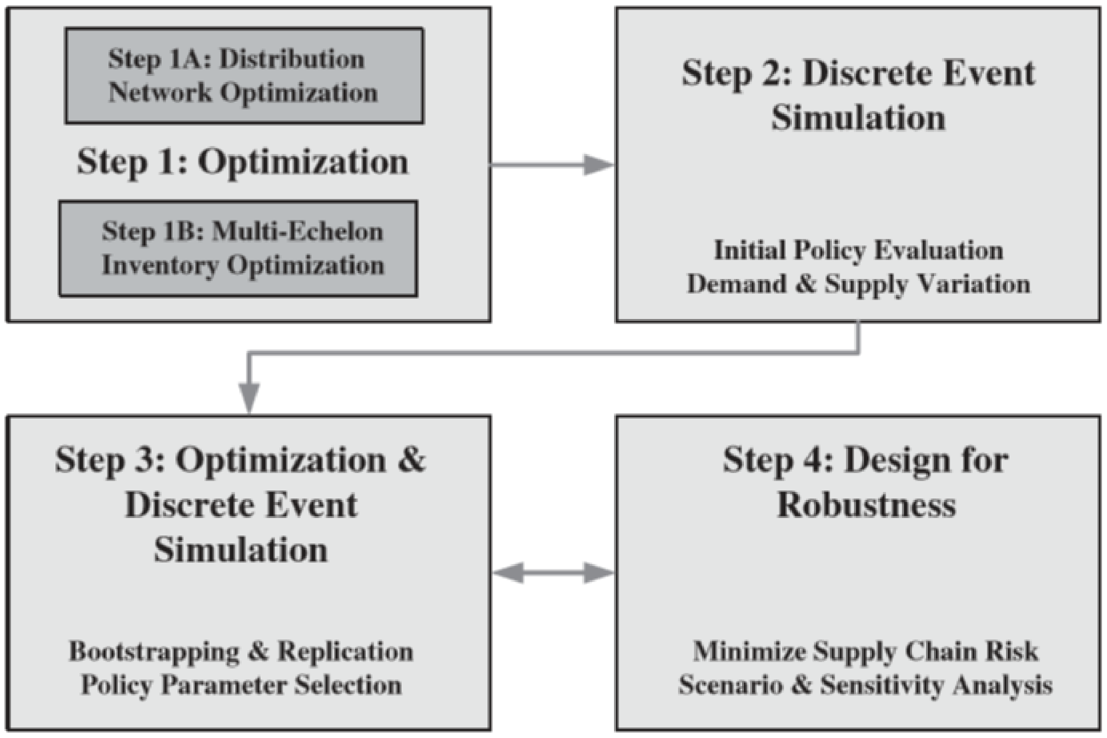

Step 1A: Distribution Network Optimization
Step 1: Optimization
Step 1B: Multi-Echelon Inventory Optimization
Step 2: Discrete Event Simulation
Initial Policy Evaluation
Demand & Supply Variation
Step 3: Optimization & Discrete Event Simulation
Bootstrapping & Replication
Policy Parameter Selection
Step 4: Design for Robustness
Minimize Supply Chain Risk
Scenario & Sensitivity Analysis

Figure 2. Four-step methodology for SO concept development.

Source: [8] S. Kumar and D. A. Nottestad, "Suppy Chain Analysis Methodology- Leveraging Optimization and Simulation Software," *OR Insight*, vol. 26, no. 2, pp. 87–119, 2012.

Other SO examples in literature include work dealing with value network design problems in the chemical industry [10]. The SO construct within this reference is seen in Figure 3. We mentioned previously the need to use multiple optimization problems. The figure demonstrates a possible example of this. The optimization model is dedicated to only solving a series of smaller sub-problems. The model used in this instance includes a series of loops. The scenario based outer loop provides the overall input variables for the value network. The two inner loops involve the DES. The second loop being the overall time period and the innermost loop being broken into smaller time periods. Within each of these planning periods, linear programming and genetic algorithm based scheduling produce a feasible product-equipment allocation and production plan [10]. A fourth Monte-Carlo loop is applied with the overall time period in which customer demand fluctuates. The final SO example we provide is shown in Figure 4. This model was also developed to support chemical supply chain networks [11]. It is similar to the model just explained with the exception that the optimization occurs on the outer loop and exit criteria is specifically outlined. The SO construct designed in this thesis takes into account similar approaches of both models.

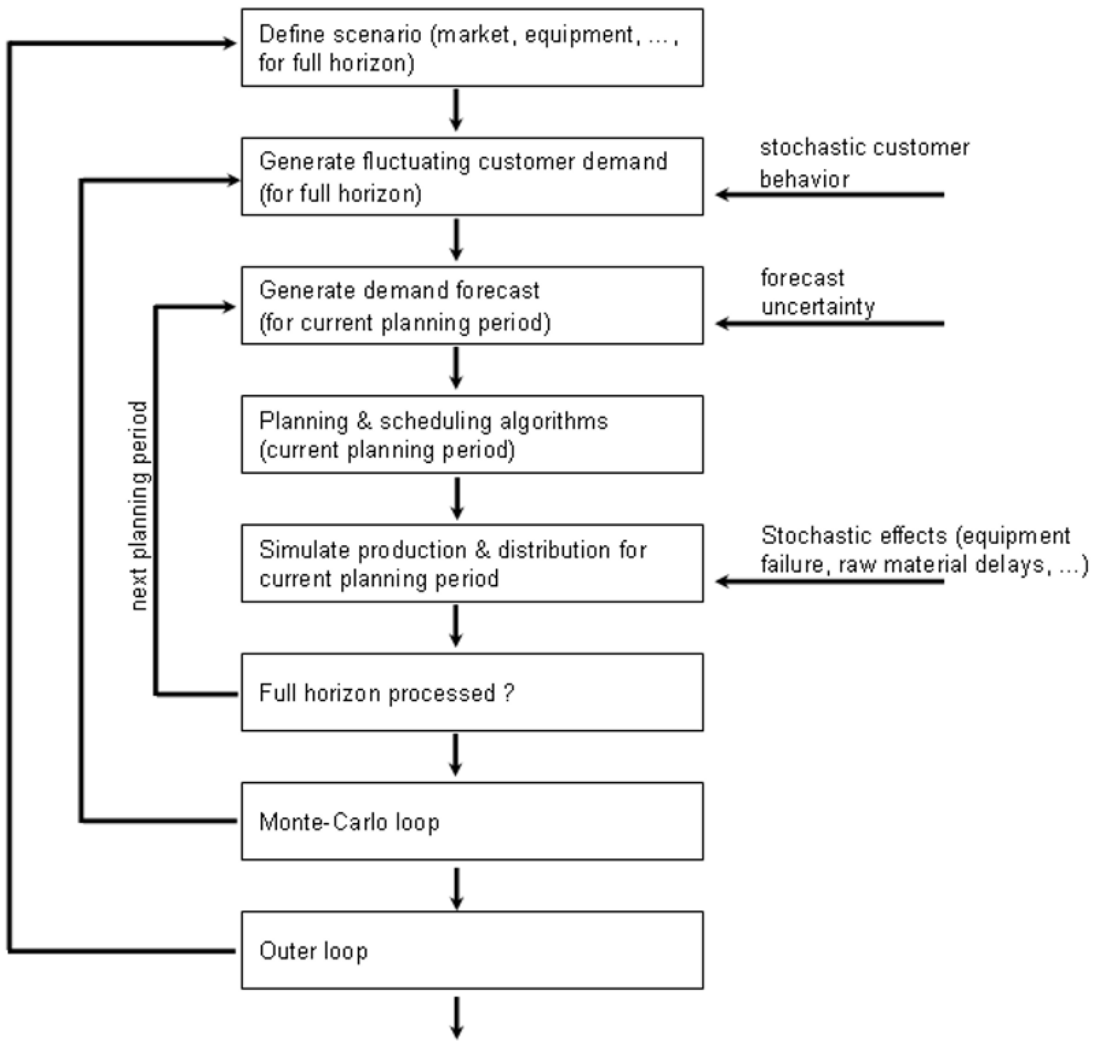

Define scenario (market, equipment, ... , for full horizon)
Generate fluctuating customer demand (for full horizon)
stochastic customer behavior
Generate demand forecast (for current planning period)
forecast uncertainty
Planning & scheduling algorithms (current planning period)
Simulate production & distribution for current planning period
Stochastic effects (equipment failure, raw material delays, ...)
next planning period
Full horizon processed ?
Monte-Carlo loop
Outer loop

Figure 3. SO construct example one.

Source: [10] M. Schlegel, G. Brosig, A. Eckert, M. Jung, A. Polt, M. Sonnenschein and C. Vogt, "Integration of Discrete-Event Simulation and Optimization for the Design of Value Networks," *Computer Aided Chemical Engineering*, vol. 21, pp. 1955–1960, 2006.

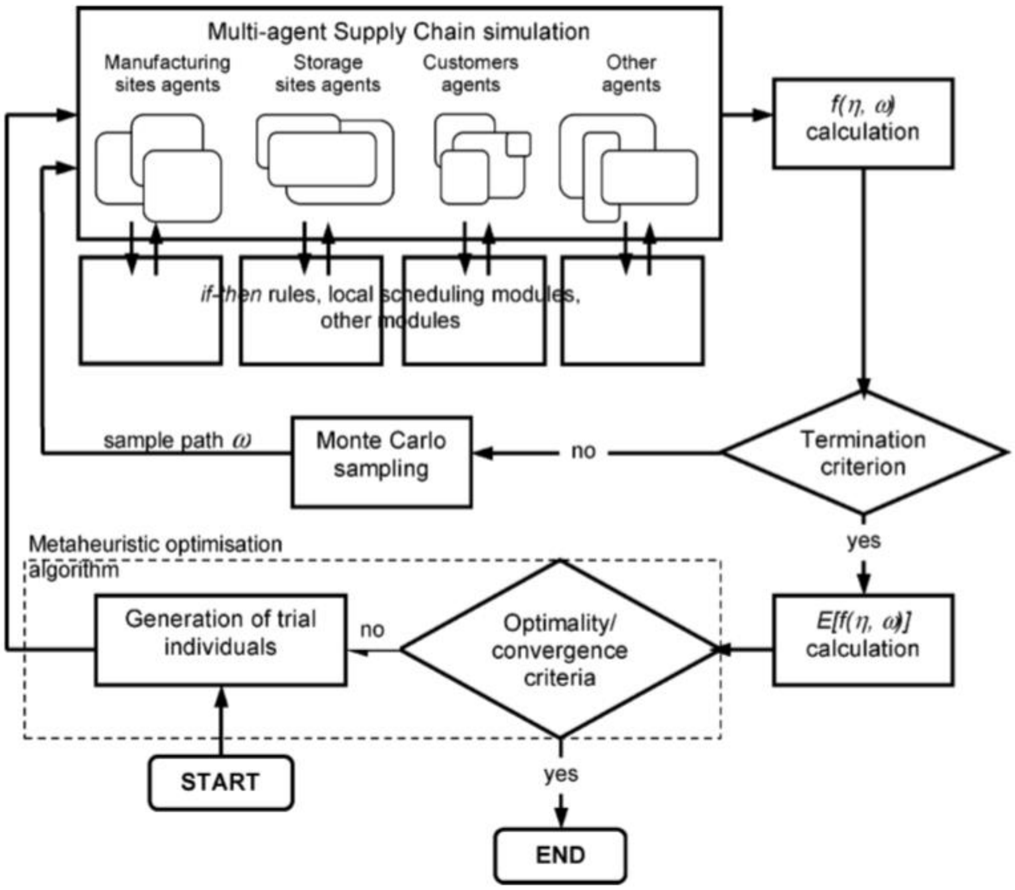

Multi-agent Supply Chain simulation
Manufacturing sites agents
Storage sites agents
Customers agents
Other agents
if-then rules, local scheduling modules, other modules
f(η, ω) calculation
Termination criterion
no
Monte Carlo sampling
sample path ω
yes
E[f(η, ω)] calculation
Metaheuristic optimisation algorithm
Generation of trial individuals
no
Optimality/ convergence criteria
yes
START
END

Figure 4. SO construct example two.

Source: [11] F. D. Mele, G. Guillen, E. Antonio and L. Puigjaner, "A Simulation-Based Optimization Framework for Parameter Optimization of Supply-Chain Networks," *Industrial and Engineering Chemistry Research*, vol. 45, no. 9, pp. 3133–3148, 2006.

Based on the research conducted, there is sufficient evidence to suggest SO techniques are beneficial in solving supply chain network problems [7]–[11]. Correspondingly, demand is sufficient to support development of commercial supply chain management simulation and optimization software solutions [12]–[14]. This includes software made by Llamsoft, Capterra, and AnyLogic.

It should be noted that MDMC has been the subject of research aimed at developing analytical tools to help improve plant operations. One such study conducted in 2009 by Northrop Grumman used linear programming techniques for this purpose [15]. The model developed within this study "calculates the 'optimal' depot-level maintenance capacity and the effect of changing the number of work positions, workload priorities, workload requirements, and/or shifts on the optimal capacity" [15]. The overall objective of the model is to minimize the difference between workload requirements and available workload requirements and available workload given a set time period. While the study addressed utilizing DES via commercial software applications (ARENA and ExtendSim) to create a management tool, the idea was dismissed due to the additional modeling requirements of a DES. In addition, adding a DES to the concept design did not fall within the requirements Northrop Grumman was tasked to accomplish [15].

THIS PAGE INTENTIONALLY LEFT BLANK

II. METHODOLOGY

This chapter focuses on providing an explanation of what optimization modeling and DES are. As will be discussed in more detail, optimization modeling provides a mathematical prescription to the problem of maximizing or minimizing a function subject to a set of constraints. However, it may not take into account or portray the detailed nonlinear stochastic intricacies that exist within the modeled system. Because of this, it can be useful to utilize simulation in order to "play out" the results provided by an optimization model. It is also important to stress the important contribution optimization provides to a simulation. The relationship between an optimization model and a DES is discussed in this chapter.

A. OPTIMIZATION MODELING

For a basic understanding of how optimization problems are developed, it is important to understand the element composition of such problems. The necessary elements are input data, decisions variables, constraints, and the objective function. Over the next few sections, the aforementioned elements are described with CPIOM provided as a test case example.

1. Decisions

The decisions in an optimization model are commonly referred to as the variables or more formally decision variables. These variables are essentially the unknowns that an optimization model is trying to determine in order to achieve the most favorable objective of the problem. Decision variables are often represented with mathematical symbols such as $X_1, X_2, ..., X_n$. A wide range of decisions can be represented by decision variables. In the case of CPIOM, the primary key decision variables are how many items of each NIIN type to stock. However, there are other control variables required in the model. These variables are not required in the DES and therefore not discussed within this thesis.

2. Constraints

The constraints of an optimization problem are simply defined as the limitations or bounds confining the model. Generally, constraint relationships are used by bounding functions of decision variables to a certain value b as follows:

A less than or equal or equal to constraint: $f(X_1, X_2, ..., X_n) \leq \mathrm{b}$

A greater than or equal to constraint: $f(X_1, X_2, ..., X_n) \geq \mathrm{b}$

An equal to constraint: $f(X_1, X_2, ..., X_n) = \mathrm{b}$

As will be demonstrated through CPIOM, optimization problems will often contain many constraints depending on the complexity of the problem [16]. In addition, other constraints may require that some or all of the decision variables be restricted to take integer values.

3. Objective

In an optimization problem, the objective function identifies some function of the decision variables in which the objective function is either maximized (MAX) or minimized (MIN). The general format of an objective function is as follows:

MAX (or MIN): $f(X_1, X_2, ..., X_n)$

4. Optimization Model Format

An example of an optimization model is represented as follows:

MAX (or MIN): $f_0(\mathrm{x}_1, ..., x_n)$ (2.1)

Subject to: $f_1(\mathrm{x}_1, ..., x_n) \leq \mathrm{b}_1$ (2.2)

$x_1, ..., x_n \geq 0$ (2.3)

The above representation reflects the objective function (Equation 2.1) that will be maximized (or minimized). The variables are subject to constraints (Equations 2.2 and 2.3). Of course, other constraints can exist depending on problem complexity.

5. CPIOM Explained

This section provides an abbreviated explanation of the CPIOM model [1]. A detailed explanation is provided within Appendix A. When formulating an optimization problem, it is important to gather and define the required input data for the model. This will provide the user with a reference guide of all indices, index sets, and parameters used. From [1], the input data for CPIOM are as follows:

$i \in I$	critical parts, also known as NIINs
$k \in K_i$	index for probability levels for part *i*
$v \in V$	vehicle types or PEIs
$v_i \in V$	vehicle type that has part *i*
n_v^V	number of vehicles type *v*
n_i^I	number of parts *i* in each v_i vehicle
n_i	total number of parts *i*. Calculated as $\sum_{v \in V_i} n_v^V \, n_i^I$
b	budget for safety stock level
c_i^{SO}	cost of each stockout of part *i*
c_i^{SS}	cost of each part *i* in safety stock (unused inventory)
q_i^0	initial stock of part *i*
δ^{q^0}	one if the initial stock (q^0 vector) counts against safety stock budget, and zero otherwise
d_{ik}, p_{ik}	demand for level *k*, and probability for that level, for item *i*:

The decision variables are as follows:

Q_i	quantity ordered for part *i*

Z_{ik}^{SO} ancillary variable for stock-out of part *i*

Z_{ik}^{SS} ancillary variable for parts *i* in safety stock that apply to the calculation of budget being used

The decision variables Q_i must be whole number positive integers. As a result, the CPIOM is considered a mixed-integer problem. Now that the decision variables are provided, the objective function and constraints are added. The overall formulation for CPIOM is as follows:

$$\text{MIN:} \quad \sum_{i \in I} \sum_{k \in K_i} p_{ik} c_i^{\text{SO}} Z_{ik}^{\text{SO}} \tag{2.5}$$

subject to:

$$Z_{ik}^{\text{SO}} \geq d_{ik} - (q_i^0 + Q_i) \quad \forall i \in I, k \in K_i \tag{2.6}$$

$$Z_{ik}^{\text{SO}} \geq 0 \quad \forall i \in I, k \in K_i \tag{2.7}$$

$$Z_{ik}^{\text{SS}} \geq q_i^0 \delta^{q^0} + Q_i - d_{ik} \quad \forall i \in I, k \in K_i \tag{2.8}$$

$$Z_{ik}^{\text{SS}} \geq 0 \quad \forall i \in I, k \in K_i \tag{2.9}$$

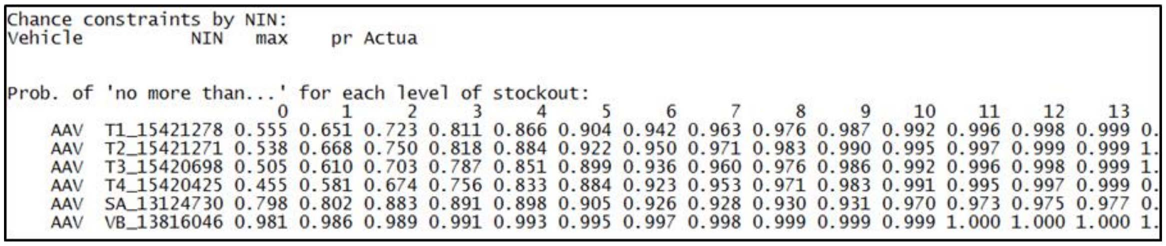
$$\sum_{i \in I} \sum_{k \in K_i} p_{ik} c_i^{\text{SS}} Z_{ik}^{\text{SS}} \leq b \tag{2.10}$$

$$Q_i \geq 0 \text{ and integer} \quad \forall i \in I \tag{2.11}$$

For the purpose of this thesis, the goal of the objective function (Equation 2.5) is to prescribe the order quantities in order to minimize expected stock-outs subject to the constraints (Equations 2.6-2.11).

A basic description of each constraint is as follows: Equations 2.6 and 2.7 calculate the stock-outs for NIIN *i* at every demand level *k* given order quantity Q_i. Equations 2.8 and 2.9 calculate the safety-stock for NIIN *i* at every demand level *k* given order quantity Q_i. Equation 2.10 limits the expected safety stock by a budget of *b*.

A key point to make about optimization models is that they are capable of accommodating elements of randomness. As seen within the CPIOM input variables, there is one variable that involves probabilities. A key feature of CPIOM centers on this variable for formulating the probability of a certain NIIN to result in a not more than a certain stock-out at varying levels. This iterative formulation over varying demand levels results in the output data displayed in Figure 5. This reflects the probabilities of no more than a certain level of stock-outs for a particular NIIN. For example, for NIIN ending in 1278, the probability of having no more than two stock-outs is 72.3 percent. While CPIOM portrays the stochastic nature of a given problem, only one answer for a set of given inputs is provided. A more detailed explanation of this formulation can be found in Appendix A.



```
Chance constraints by NIN:
Vehicle          NIN   max    pr Actua


Prob. of 'no more than...' for each level of stockout:
                           0     1     2     3     4     5     6     7     8     9    10    11    12    13
    AAV  T1_15421278 0.555 0.651 0.723 0.811 0.866 0.904 0.942 0.963 0.976 0.987 0.992 0.996 0.998 0.999 0.
    AAV  T2_15421271 0.538 0.668 0.750 0.818 0.884 0.922 0.950 0.971 0.983 0.990 0.995 0.997 0.999 0.999 1.
    AAV  T3_15420698 0.505 0.610 0.703 0.787 0.851 0.899 0.936 0.960 0.976 0.986 0.992 0.996 0.998 0.999 1.
    AAV  T4_15420425 0.455 0.581 0.674 0.756 0.833 0.884 0.923 0.953 0.971 0.983 0.991 0.995 0.997 0.999 0.
    AAV  SA_13124730 0.798 0.802 0.883 0.891 0.898 0.905 0.926 0.928 0.930 0.931 0.970 0.973 0.975 0.977 0.
    AAV  VB_13816046 0.981 0.986 0.989 0.991 0.993 0.995 0.997 0.998 0.999 0.999 0.999 1.000 1.000 1.000 1.
```

Figure 5. This sample output probability distribution for the CPIOM reflects the chance a certain NIIN will have no more than a certain stock-out.

Source: [1] J. Salmeron, A. Buss, T. Curling and M. Kress, "Plant utilization at Marine Corps Logistics Command," Naval Postgraduate School, Monterey, 2014.

B. DISCRETE EVENT SIMULATION

The purpose of this section is to provide a general understanding of DES and how it will be used in the context of this thesis. As the name implies, a DES is a simulation in which interactions within a system occur as specific discrete events in time. Over the next few sections a description of how an event is defined, when an event is executed, and what occurs during event execution in the context of this thesis will be provided. Specifically, this section will discuss the primary elements of a DES. These elements are states, events, and scheduling relationships.

1. States

The primary goal of a DES is to model the changes that occurs within a system or process for a certain attribute or combination of attributes over time [17]. Capturing this change is the cornerstone function of a DES in regards to analyzing a process or system. The attribute(s) being modeled are referred to as the *state variable(s).* The collection of all the state variables in the simulation is referred to as the *state space*. The overall combined status of the state space at any given moment in time constitutes the simulation's *state* [17]. A simple way to think of this is to consider the simulations state as the overall condition of the system at a certain point in time. It is critical to understand and identify the state space of a system or process, because it provides the primary basis in which the system is to be modeled. In developing a DES, the primary focus is on those parts of the system that have an effect on the simulations state. The mechanics of how a DES keeps track of a simulation's state will be detailed in Chapter III.

2. Events

Every process consists of key actions that occur at certain points in time that change the state of the overall system. These points in time are known as *events* [17]. When an event occurs within a system, the action(s) taken within that event will affect the *state* of the system. An event may reference information about objects, also known as entities, which must be passed along within the simulation. Some common examples of entities include customers, passengers, and vehicles. As alluded to previously, one of the key points to understand when discussing DES is the method in which events occur through simulation time. Unlike a time-stepped approach that produces interactions in regular intervals or steps, simulated time in a DES moves according to the time of the next scheduled event. These pending events are maintained chronologically as *event notices* in a list known as a *future event list* [17]. As the simulation progresses, event notices are removed from the future event list as their respective execution time is reached. Correspondingly, new event notices are also added to the list according to the simulation design. In some cases, event notices can even be canceled.

When an event is reached according to the future event list, the event triggers the actions defined by that event. There are several actions a state will implement: Two fundamental actions include inducting changes to state variables (also known as *state transitions)* and establishing a *scheduling relationship* between events. Other important actions include updating entity information and tracking statistical data.

3. Scheduling Relationship and Time Advance

Establishing a *scheduling relationship* between events is the driving catalyst of the simulation [17]. Every event that occurs is assigned some specific action affecting the simulation state and often times scheduling another event. The point at which events are no longer created represents the time at which the simulation will end. This can be best explained using Figure 6. When the simulation begins, an initial event is scheduled and the event notice is placed on the future event list. The simulation advances to the first scheduled event referenced on the event list and removes that event notice from the list. The referenced event will then execute the actions assigned to it. If one of the actions of the event includes scheduling another event(s), they will be placed onto the future event list as event notices. The simulation will then advance to the next scheduled event until the exit criteria of the DES is achieved or the future event list is empty. The topic of exit criteria will be discussed further in Chapter III. This approach to simulation is computationally efficient as the program only needs to process events, as they are required.

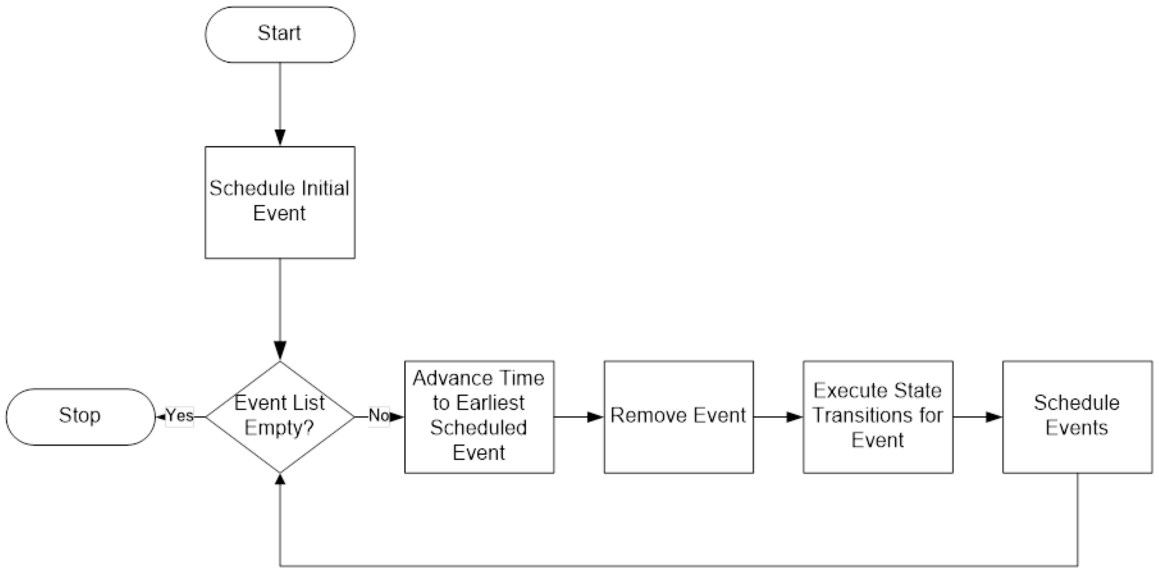


Figure 6. This graph depicts the algorithm within a DES of processing events on an event list.

Source: [17] A. Buss, "Discrete event simulation modeling," unpublished.

4. Simulation Parameters

The simulation parameters of a DES are those variables that do not change during the course of a replication of the simulation [17]. Simulation parameters are in many cases synonymous with the constraints of an optimization model. For example, a system will have certain number of employees available, workstations available, or total number of vehicles entering the system. However, this is not always the case. There may be scenarios where the DES model does not contain a constraint that exists within the optimization model. For example, the number of available work bays is an important parameter within the DES outlined in Chapter III and not found in the CPIOM. This converse situation may also occur.

5. Event Graphs

Event graphs are an important modeling tool used in developing a DES. Using event graphs allows the simulation developer to organize and visualize the process being simulated. This is accomplished by essentially copying the laydown of the events as they appear in the real world. As explained in [17], event graphs consist of nodes to represent events and directed edges to represent scheduling relationships. Figure 7 is an example of a simple event graph extracted from [17] for a multiple server queue. In this example the nodes represent the events and the directed edges represent the scheduling relationships between events. From Figure 7, there are a total four events: Run, Arrival, StartService, and EndService. The expression beneath each event reflects the state transitions taken for that event. From Figure 7, the state variables are Q , the number of customers in the queue, and S, the total number of available servers. The Run event initializes Q to 0 and S to the parameter k. The directed edge from the Run event to the Arrival event means that the Run event schedules the Arrival event t_A time units after the Run event. The self-scheduling arc on the Arrival event means that it schedules another Arrival event t_A time units in the future. The sequence $\{t_A\}$ is considered a parameter of the model, and represents the successive customer interarrival times. This sequence $\{t_A\}$ can either be a pre-specified collection of numbers or generated by a probability distribution.. Each occurrence of the Arrival event will therefore carries out three separate actions. First, it will increase the state variable value by one (the state transition). Second, it will schedule another arrival event. Third, it will attempt to schedule the start service event. Along the scheduling edge there is an annotation with a (S > 0) above it. This annotation means that a certain condition must be met in order to schedule the respective event. The condition (S > 0) means that a server must be available for the start service event to be scheduled. Similarly, the condition (Q > 0) on the edge from EndService to StartService means that at least one customer must be in the queue for a StartService event to be scheduled. In summary, an event graph provides the structure for DES program implementation.

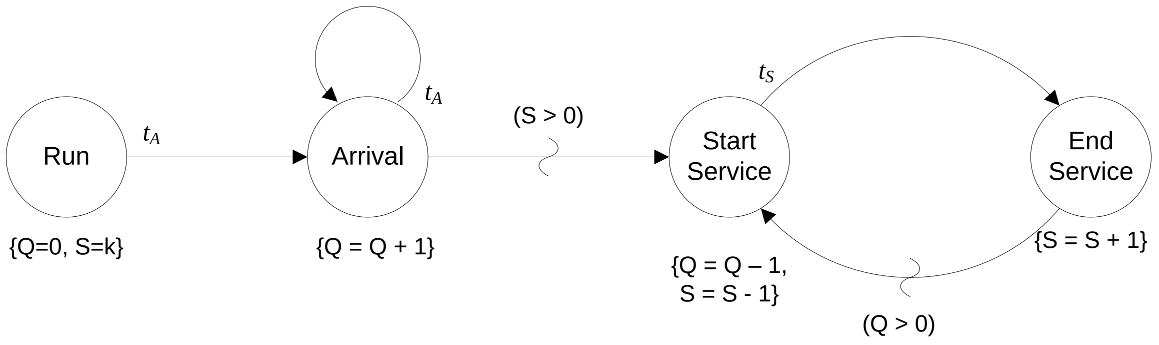


Figure 7. Event Graph for multiple server queue with citation

Source: [17] A. Buss, "Discrete event simulation modeling," unpublished.

6. Entities

Entities can be interpreted as the objects that move through a process or system. Examples of entities include job orders, customers, or vehicles. The use of entities is convenient when the modeled process must know what attribute(s) the entity possesses. For example, there may be several production lines modeled in which multiple vehicles are produced. Each vehicle entity will have an attribute identifying what type of vehicle it is. When the vehicle enters the system, the DES will use this attribute to determine what production line to enter. A vehicle entity may include inventory attributes. An inventory attribute will enable the DES to determine what parts each individual vehicle requires. In order to leverage the full analytical capability of a DES, entities may also include attributes involving time. This includes the creation time and the last time an entity encounters significant events. The attributes used are dependent on organizational objectives and complexity of the represented system. Entities will be discussed in more detail within Chapter III.

C. JOINT OPTIMIZATION AND SIMULATION AND MODELING

An intrinsic link is developed when combining Optimization modeling and DES to analyze a particular process. As discussed previously, optimization modeling will allow a deterministic, best case scenario to be calculated. A DES on the other hand can account for the internal stochastic interactions of a process otherwise not possible using optimization modeling. If designed properly and assuming the probabilistic data used is representative of the future, a simulation can accurately replicate the process being analyzed. This section will discuss the important role optimization modeling and DES play for each other. This includes a discussion on how the techniques facilitate model verification. Lastly, this section will provide a case study using simulation to enhance the CPIOM results.

1. Optimization and DES Intrinsic Link

By now, it should be recognized that an optimization model and DES naturally complement each other. Specifically, an optimization model's prescriptive and static nature supplements the descriptive and dynamic nature of the DES and vice versa. Seeing that the two techniques are mutually supporting of each other in order to achieve a decision making objective, an intrinsic link between them should exist.

The first reason is that both techniques require identical inputs since they are modeling the same process. The second reason is that the output of the optimization model provides important input information to the DES and vice versa.

A DES only "replays" the process in accordance with the inputs it is provided. Without some analytical approach such as a design of experiments or optimization modeling, trying to determine a best-case scenario using DES alone is a severely inefficient method. For example, assume there is a group of widgets needing to be repaired that require four different NIINs, each having a different cost. For each NIIN, a quantity of *n* = 15 parts is required. In order to determine the cost of every combination of the r = 4 NIINs using simulation, a total of 1,365 possible combinations would need to be simulated:

$$ {}_{n}C_{r} = \frac{n!}{r!(n-r)!} = 1,365 \tag{2.12} $$

While it may be possible to run an algorithm that simulates every possible combination, this is certainly a poor use of computing resources when optimization modeling can determine an optimal solution mathematically. Assuming the 1,365 simulations run a total of 10,000 repetitions per simulation, the use of an optimization model would reduce the number of simulation repetitions by 13,640,000. This is just a simple illustration of why optimization modeling is important when utilizing DES in the context of this thesis.

While optimization modeling is a very powerful tool, its deterministic nature is also a potential limitation. As mentioned previously, an optimization model can take probabilistic input data to produce a mathematical result such as represented in Figure 5. However, optimization is not dynamic in nature and therefore limited in its ability to analyze the internal stochastic intricacies of a process, which are naturally handled by a DES.

Every simulation repetition run potentially has a different outcome. This brings up a third primary reason an intrinsic link exists: Without a DES, an optimization model would essentially be unable to compensate for the stochastic and dynamic nature of the modeled process. A DES allows an optimization model to account for this stochastic nature by facilitating "course corrections" within the process as time progresses. This is accomplished by allowing the optimization model to essentially take a snapshot of the simulations state at prescribed times in order to re-optimize the process being simulated. The specifics of how this is accomplished in regards to CPIOM is explained in Chapter III.

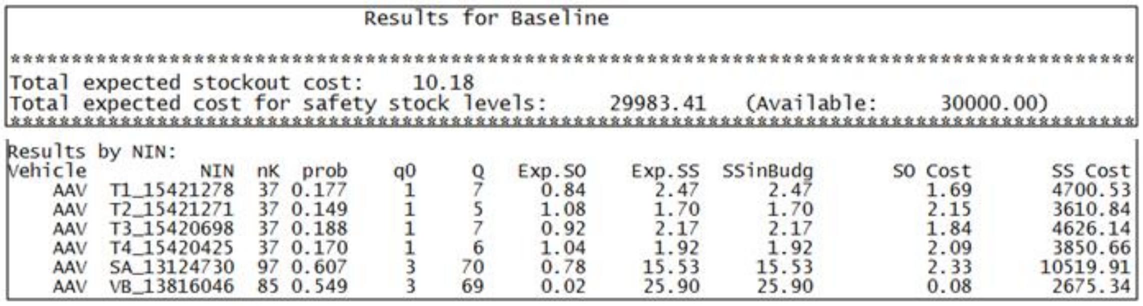
Lastly, an intrinsic link exists because a DES provides additional statistical information other than the optimization models bottom-line results. Additional information detailing the stochastic intricacies of a system not only facilitates the model verification process but also the decision making process for which the optimization model is designed. The next section demonstrates this concept of integrating simulation and CPIOM.

2. Monte Carlo Simulation with CPIOM

This section will briefly show how simulation can provide additional insights for an optimization model. For this demonstration, the output data of CPIOM [1] is analyzed using a Monte Carlo simulation technique. The use of a Monte Carlo simulation is a basic process in which CPIOMs input data and the optimal order quantity output is used to calculate total safety stock and total stock-out cost. For each calculation, the demand is generated from the parts distribution probability input file. As a result, each calculation will be different. In this example, the simulation is run a total of one million replications using the optimal order quantity data provided by CPIOM for the baseline scenario listed in [1]. The results provided by this simple demonstration reflect how simulation can not only support the model verification process, but also provide additional statistical insight.

a. Model Cross Verification

Cross verification between a simulation and optimization model allows the program developer to ensure the program is in fact working as intended. Verification is an important step in the model validation process as it ensures the technical details are being met. Perhaps one of the most difficult tasks when developing a model is being able to validate that the model is accurately representing the real-world system. This validation cannot happen unless we know that the model is first verified to be accurate. The utility of using simulation and optimization to support this cross verification process is seen when looking at the baseline scenario output data from [1]. The CPIOM's output data for the baseline scenario is seen in Figure 8 [1]. The optimization output provided is the bottom line result of the optimization model. That is the total expected stock-out cost of \$10.18 and total safety stock cost of \$29,983.41. As seen in Figures 9 and 10, the Monte Carlo simulation output data for the baseline scenario yields very similar metrics. Over a million repetitions, the simulation reflects a mean total stock-out cost of \$10.19 and a mean total safety stock cost of \$29,967.92. The closely matched metrics provide cross verification between both the simulation and the optimization model. By ensuring the simulation and optimization are functioning as programmed, future iterations of either model can be developed with greater degree of confidence.



```
                              Results for Baseline

*************************************************************************************************
Total expected stockout cost:    10.18
Total expected cost for safety stock levels:     29983.41    (Available:      30000.00)
*************************************************************************************************
Results by NIN:
Vehicle          NIN  nK  prob    q0      Q   Exp.SO    Exp.SS  SSinBudg        SO Cost        SS Cost
    AAV  T1_15421278  37 0.177     1      7     0.84      2.47      2.47           1.69        4700.53
    AAV  T2_15421271  37 0.149     1      5     1.08      1.70      1.70           2.15        3610.84
    AAV  T3_15420698  37 0.188     1      7     0.92      2.17      2.17           1.84        4626.14
    AAV  T4_15420425  37 0.170     1      6     1.04      1.92      1.92           2.09        3850.66
    AAV  SA_13124730  97 0.607     3     70     0.78     15.53     15.53           2.33       10519.91
    AAV  VB_13816046  85 0.549     3     69     0.02     25.90     25.90           0.08        2675.34
```

Figure 8. CPIOM output results for the baseline scenario.

Source: [1] J. Salmeron, A. Buss, T. Curling and M. Kress, “Plant utilization at Marine Corps Logistics Command,” Naval Postgraduate School, Monterey, 2014.

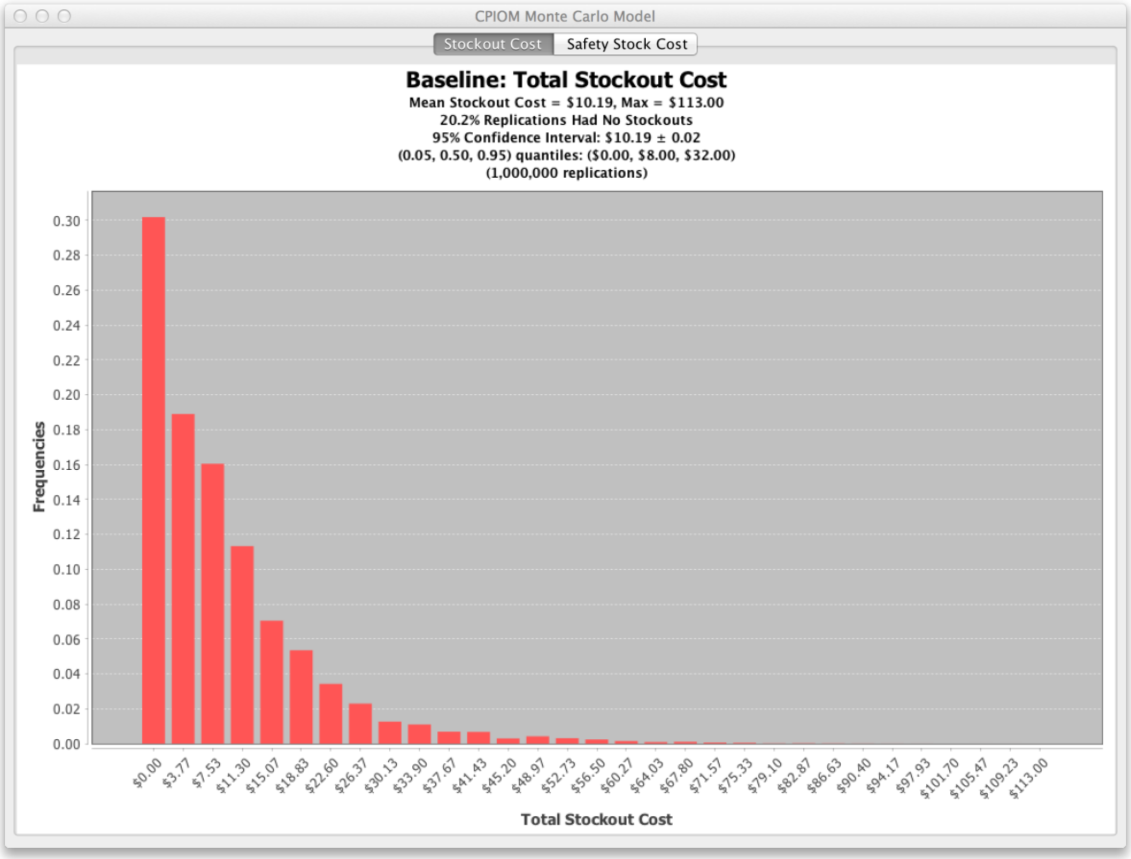
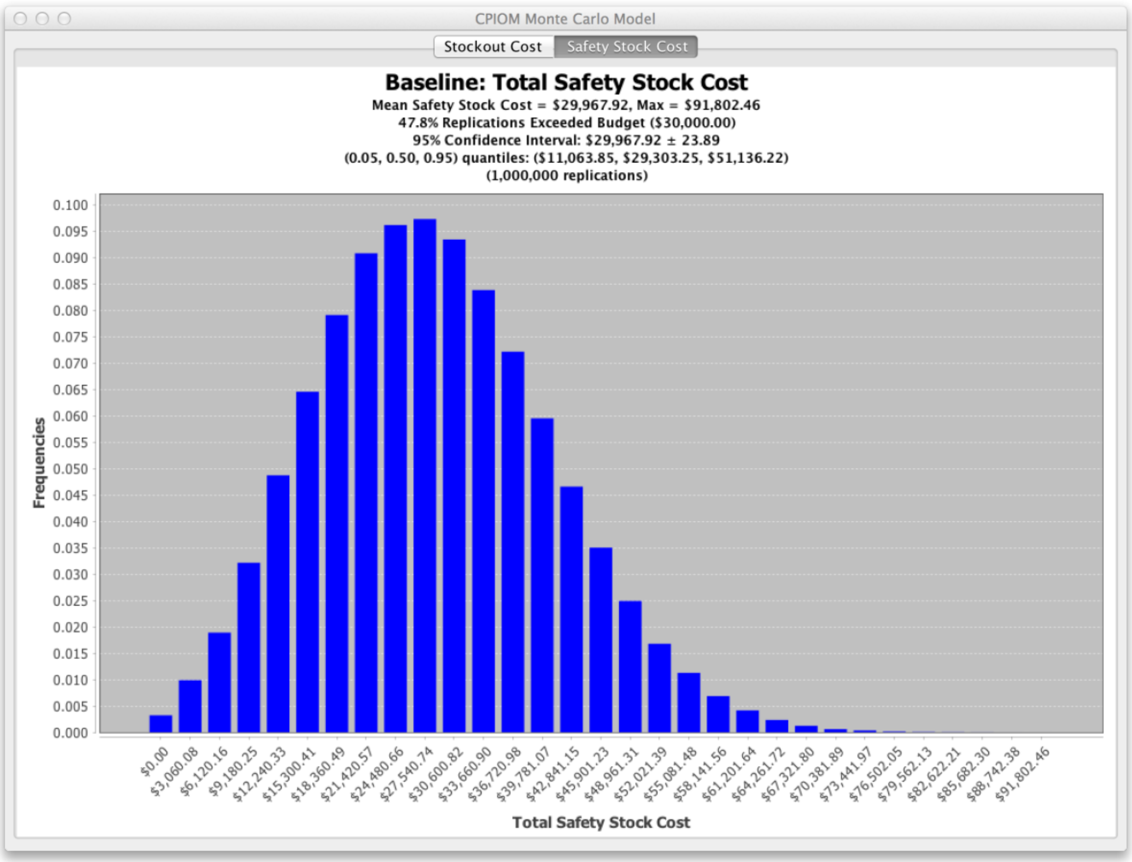


Figure 9. From [1], Monte Carlo simulation output for total stock-out cost for the baseline scenario.

Source: [1] J. Salmeron, A. Buss, T. Curling and M. Kress, “Plant Utilization at Marine Corps Logistics Command,” Naval Postgraduate School, Monterey, 2014.


Baseline: Total Safety Stock Cost

Mean Safety Stock Cost = $29,967.92, Max = $91,802.46

47.8% Replications Exceeded Budget ($30,000.00)

95% Confidence Interval: $29,967.92 ± 23.89

(0.05, 0.50, 0.95) quantiles: ($11,063.85, $29,303.25, $51,136.22)

(1,000,000 replications)

Figure 10. Monte Carlo simulation output for total safety stock cost for the baseline scenario.

Source: [1] J. Salmeron, A. Buss, T. Curling and M. Kress, "Plant utilization at Marine Corps Logistics Command," Naval Postgraduate School, Monterey, 2014.

b. *Improved Statistical Analysis*

Because the simulation is able to provide dynamic run-by-run analysis not possible with a static optimization model, additional statistical analysis can be achieved. As seen in Figures 9 and 10, the multiple runs allows for the data to be used in a histogram according to total stock-out cost and total safety stock cost. This visual representation may facilitate statistical analysis thereby leading to improved decision making otherwise not possible using an optimization model alone. When looking at the statistical data resulting from the Monte Carlo simulation, a couple of useful pieces of information can be deduced. Notice from Figure 9 that 20.2 percent of replications will result in no stock-out cost and about 30 percent will result in none or very little stock-out cost. In other words, over 70 percent of replications reflect a stock-out cost using the "optimal" CPIOM output data. When looking at the total safety stock cost metric in Figure 10, the simulation reflects 47.8 percent of replications exceeding the prescribed budget. In the operational environment, exceeding the budget is often unacceptable. Both of these cases demonstrate how simulation will allow managers to more readily answer questions concerning risk and ultimately shape their decisions. It is noted that optimization models are not without useful statistical information. For example, an extended version of CPIOM reflects the chance constraints by individual NIIN. As discussed earlier, Figure 1 provides the likelihood for a particular NIIN to have no more than a certain number of stock-outs. In conclusion, this co-optimization modeling and simulation building approaches work complementarily to each other thereby allowing for improved statistical analysis.

THIS PAGE INTENTIONALLY LEFT BLANK

III. IMPLEMENTATION

One of the primary objectives of this thesis is to design, develop, and implement a program that establishes the initial framework for a more robust Order Management Tool (OMT). The OMT references the overall program discussed in the remaining chapters. As an initial construct and proof of concept, the underlying DES model within OMT is set up to accommodate the minimal key events of an MDMC production line. Future iterations of OMT can be expanded, tested, and evaluated in accordance with the live environment.

The foundational implementation objective is to design OMT in a manner that allows seamless integration of the DES and CPIOM. In order to allow for MDMC to easily experiment with and validate the program, input data was in the form of Excel spreadsheets, which are currently used and easily understood by MDMC personnel.

The secondary objectives include providing the user with output data in a graphical format and using open-source solutions. The graphical output component of the program will allow the user to quickly visualize important statistical data. In regards to open-source solutions, this provides MDMC with uninhibited access to the program resulting in increased flexibility in program development without the burden of contracted support.

This chapter will discuss in detail how these requirements were implemented. This includes introducing the tools used as well as potential alternative tools. The joint DES and optimization construct will then be explained in order to provide the overarching implementation concept of the system. With an understanding of the joint construct, the DES model will be explained in detail. Finally, this section will walk through aspects of the computer code in order to provide an understanding of program implementation from a computer programming perspective. As mentioned earlier, implementation of CPIOM is not the focus of this thesis. CPIOM will only be discussed as it pertains to OMT programming implementation.

A. DEVELOPMENT TOOLS

In this section, the various resources used for OMT development are explained. The DES portion of OMT uses open-source resources obtained via the World Wide Web. By utilizing these open-source resources, this allows for MDMC and LOGCOM to have full access and control to the programming code. The only limitation that may exist is the ability to install the General Algebraic Modeling System (GAMS) onto government-owned computers [18].

1. GAMS and Cplex

GAMS is a commercial off-the-shelf program specifically designed for modeling linear, nonlinear, and mixed-integer optimization problems. The program is especially useful when dealing with large complex problems containing many variables and constraints. The features of GAMS allow the user to focus on modeling rather than the technical machine-specific problems.

GAMS/Cplex (www.gams.com/dd/docs/solver/cplex/) is a solver that allows users to combine the high level modeling capabilities of GAMS with the power of Cplex optimizer [19]. This is designed to solve large-scale optimization problems employing state-of-the-art solution algorithms designed for linear and mixed-integer programming. Because GAMS is used to implement CPIOM, the computer program developed in this thesis currently interfaces with GAMS as opposed to other alternatives. It is important to note that comparing between GAMS/Cplex with the open-source solver LPSolve (described next) showed GAMS/Cplex to be a substantially faster solution engine.71

2. LPSolve

LPSolve (http://sourceforge.net/projects/lpsolve/) is one of many open-source optimization solvers [20]–[22]. This solver is based on the revised simplex method and the branch-and-bound method for integer problems. LPSolve is being mentioned in this thesis to show that CPIOM can be implemented using only open-source solutions. The limitation is that LPSolve may not necessarily achieve the result as effectively as

GAMS/Cplex, which is used by the current OMT construct. This is most likely due to the robust capabilities of their proprietary counterparts.

3. Java Based Tools

OMT development uses four open-source Java based resources. The first resource is the integrated developer environment known as NetBeans [23]. This fully featured program enables software developers to develop Java desktop, mobile, and web applications. The second resource is an open-source library developed by Naval Postgraduate School called SimKit [5]. This library enables development of robust DES programs. The third resource is derived from the Apache POI project found at [3]. The project's mission is to create and maintain Java Application Programming Interfaces for manipulating various file formats based upon the Office Open XML standards and Microsoft's OLE 2 Compound Document format (OLE2). In short, a program can read and write Microsoft Excel files using Java. In addition, it can be utilized to read and write Microsoft Word and Microsoft PowerPoint files. The fourth resource is the Java JFreeChart library derived from [4], which is used to create the output charts for OMT.

B. JOINT OPTIMIZATION AND DES CONSTRUCT

This section explains how the optimization model and the DES interfaced. The explanation of this interface includes the specific details of how the order management process is integrated with the production process over time. In addition, a brief explanation of how the simulation terminates, followed by a basic example of the joint optimization and DES concept.

1. Implementation Overview

Implementation of the joint optimization and DES construct is a straightforward concept in which CPIOM and the DES rely on each other for critical input information as time progresses. Both the production process and the supply system are dynamic systems that are often unpredictable. Unpredictable changes result in plans and outcomes being altered. This joint concept will allow for the situation to be reassessed at designated times in order to accommodate these changes. It should be made clear that the DES is stochastic based on historical parts ordering and vehicle condition data. When running the OMT, it is assumed that the historical data used are representative of the future behavior of the supply system and vehicle condition. If this is indeed the case, this joint concept will be effective in making the small adjustments required to achieve the objective of the OMT.

The CPIOM and DES are dependent on two primary inputs. The first inputs are the initial parts inventory levels. With the exception of the initial optimization run, the critical information provided by the DES and required by the CPIOM is the current parts inventory levels. The current parts inventory levels maintained by the DES serve as the initial parts inventory levels at the moment the optimization model is run. The unknowns to the DES are the parts order quantities. This critical information is provided by CPIOM and required by the DES. As discussed in the previous chapter, the order quantities for each part are the decision variables of the optimization model. Once CPIOM is run, the DES will use CPIOM's decision variable outputs as input variables. This iterative process of the CPIOM feeding critical information into the DES and the DES feeding critical information into the CPIOM will repeat as defined by the user.

2. Termination Criterion

As illustrated in Figure 11, a complete cycle of the simulation may consist of one or many iterative loops within the OMT construct before meeting the exit criteria. The number of loops made and total time this cycle lasts is based on the optimization frequency and total simulation time as dictated by the user. Once the exit criterion is achieved, the loop will end and the cycle will repeat itself until the required overall criteria has been achieved for the simulation. For example, a user may dictate for the cycle to consist of one optimization per yearly quarter for a period of one full year. In other words, the loop will occur four times and exit directly from the DES on the last loop.

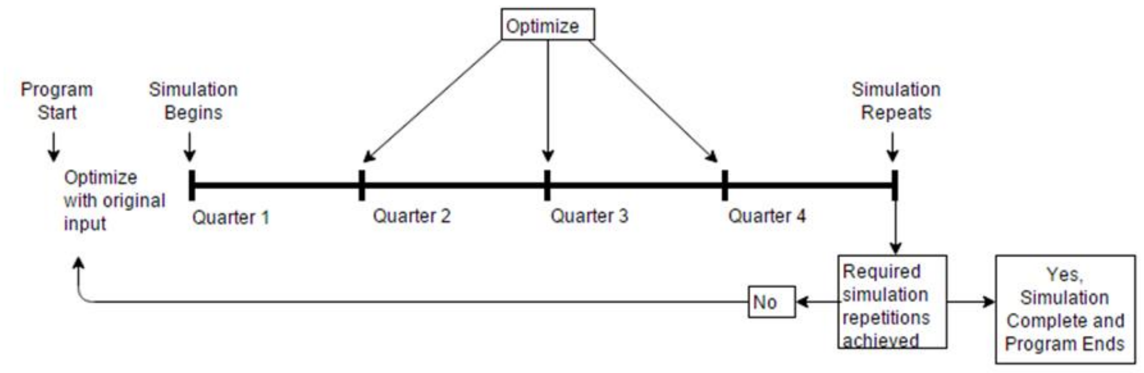

Optimize
Program Start
Simulation Begins
Simulation Repeats
Optimize with original input
Quarter 1
Quarter 2
Quarter 3
Quarter 4
Required simulation repetitions achieved
No
Yes, Simulation Complete and Program Ends

Figure 11. An example of the OMT joint construct. CPIOM is applied throughout the DES as dictated by the user.

As alluded earlier, the OMT joint construct cycle repeats until achieving the required simulation repetitions. There are a couple of methodologies in defining the overall termination criteria. The more formal way is based on convergence to some steady state using statistical methods for a specific state variable. For example, at the end of every complete cycle of the joint model, the DES could record the production completion rate. A possible convergence metric may be looking at the residual value based on the production completion rates. At the end of each joint model cycle, the residual would be recalculated and the absolute values for the difference between the newly recalculated residual value and the old residual value calculated. If there is a difference, a counter will reset itself. If not, the counter will increase. Once the counter reaches a certain threshold, the simulation will end. Essentially what is happening is that the residual is stabilizing (meaning that there is not enough variation to warrant further iterations). This value will fluctuate and eventually stabilize as the number of simulation samples increases. In other words, the simulation will converge to a certain residual value. There are several approaches that can be found in literature discussing the topic of determining the number of simulation runs to be made [24], [25]. The simpler and non-scientific approach would be to simply set a very high number of simulation repetitions as the exit criteria. This is ideal if the computing speed and available memory permit, and is the approach used in this thesis.

3. Concept Example

Figure 11 provides a simple example of the overall OMT construct process. In this example, the optimization frequency is set to a rate of one optimization per yearly quarter. The total simulation time is one full year. This means that CPIOM is run a total of four times in the course of a single replication. When the DES reaches the one year mark, the OMT program will exit the DES. This completes the first simulation replication. The OMT will compile the statistical data from the DES and determine if the exit criterion (required quantity of simulation runs) is achieved. If so, the OMT will exit out of the overall simulation and take appropriate actions prior to closing. Otherwise, the OMT will repeat the process.

Walking through Figure 11, when the OMT is initiated there is a gap between when the DES begins and when the optimization model is run. During this time, the OMT will extract data from the user-provided Excel spreadsheet and compile these data for use by both the CPIOM and the DES. Once this process is complete, the CPIOM will run. This first run of the CPIOM within the joint model construct is unique because the initial inventory was provided by the user. Once CPIOM completes its initial run, the DES will begin and the simulation time clock starts. From the onset of the simulation starting, the DES will initialize itself by extracting data from not only the previously compiled user-provided data but also from output provided by the initial CPIOM run. In addition, the DES will schedule the first optimize event a quarter in advance. The DES will then progress through the event list. Upon reaching the beginning of the second quarter, the second optimization will occur. This run and all subsequent runs of CPIOM will now use the current parts inventory maintained by the DES as its initial inventory input. The CPIOM is run and another CPIOM run is scheduled a quarter in the future. Again, the DES takes the output from the CPIOM and initiates the appropriate follow-on actions required as result of the optimization event. The CPIOM will repeat itself two more times as the DES progresses through time. Upon reaching the end of the simulated year, the DES will then exit the joint model construct. At this point, the program will compile the statistical data as well as determine if the joint model construct should be repeated. If the exit criteria or number of simulation repetitions has not been achieved, the process just described will be repeated. Of note, the extraction and compilation of user data only occurs once. The CPIOM can immediately run with the originally compiled data. If exit criteria is achieved, the OMT will not repeat the process and take all pre-closing actions before terminating.

C. OMT PROGRAM AND DES MODEL EXPLAINED

With the joint optimization and DES construct explained, we set the stage for explaining the fine details of how the DES is implemented. This section begins by providing a general overview of the scenario in order to gain an understanding of the simulated model. This will lead into a detailed discussion of each component and subcomponent within the DES, (i.e., the individual events, states, parameters, entities, and scheduling relationships). This discussion of DES components and subcomponents culminates by walking through the model being simulated using the event graph for this DES.

1. Model Scenario

The model in which this program is based is formulated upon a basic production scenario where PEIs of a single type enter into a production plant in order to be completely dissembled and then reassembled from a parts requirement perspective. The condition of each PEI is unknown upon entry. Once the PEI has been completely dissembled, a determination as to the condition of the PEI from a parts perspective is made. The PEI is then reassembled with serviceable parts. This completes the production process. Obviously, new parts will be required in order to replace the unserviceable parts. Without knowing the condition of the vehicles ahead of time, plant managers must somehow estimate what parts will be required in order to preorder the parts and avoid a stock-out situation. The plant also does not want to exhaust its budget ordering parts that will not be needed i.e., having safety stock. Large amounts of the budget tied up in safety stock could inhibit purchasing parts in a stock-out situation. The combined effect of parts existing in a stock-out and safety stock status could result in the production line being impeded and thereby reducing production plant capacity. Of course, there may be other reasons why MDMC managers are seeking ways to improve parts management.

The fundamental goal of this basic scenario is to allow a fully functional program to be developed. This will allow future developers to focus solely on validating the existing model as well as expanding the scenario. This includes, but is certainly not limited to, adding multiple PEI types, increasing individual production line fidelity, adding additional model constraints, and adding multifunctional production line capability for a specific PEI.

2. OMT Program Requirements

With the base scenario and joint optimization construct in mind, attention can be focused on determining the functional requirements of the OMT program. In addition, the primary and secondary objectives mentioned earlier in this chapter are also converted into functional requirements of the DES program. Functional specifications require that the OMT be capable of

1. extracting user input from Excel spreadsheets.
2. creating csv input files for use by CPIOM.
3. seamlessly interfacing the DES with CPIOM.
4. creating individual PEIs of a single type.
5. determining individual PEI condition using historical data from a parts perspective.
6. determining an individual part's lead time using historical ordering data.
7. maintaining an inventory of parts.
8. calculating, ordering and receiving parts.
9. maintaining simulation state statistics
10. producing graphical representation of state statistics (secondary requirement)
11. using open-source tools to the maximum extent possible (secondary requirement)

In developing the OMT, each requirement is independently tested and developed as its own separate Java class to the maximum extent possible. This approach helps in breaking the program into smaller, more workable pieces, which ultimately helps when debugging or updating certain aspects of the program.

3. DES Entities

The entities within this DES are the individual PEIs. Because the PEI class extends SimKit's Entity class, each PEI contains all of the functions and attributes associated with the Entity class. This includes the inventory, identifier, creation time, and time stamp. The PEI class in particular assigns additional attributes to include the PEIs internal inventory, time to assemble, and time to disassemble. The assembly and disassembly time are only temporary attributes. A more appropriate method in assigning these times would be to reference an index. This is especially true if the production line fidelity is increased. One production line could consist of hundreds steps resulting in a complex index of production times. As the PEI moves through the production line, the PEIs internal inventory would be adjusted as appropriate. The DES's ability of knowing what parts a PEI has and being able to remove and add parts from it is the cornerstone function of the DES.

4. DES Parameters

As mentioned earlier, the parameters are all of the input variables that do not change within the DES. This includes the following:

- Cost of each part per NIIN
- Parts required for each PEI type
- Total number of bays
- Re-optimization time
- Total number of PEIs entering system per optimization period
- Total simulation time

Some of these parameters must be placed into an array for easy reference by the program. For example, the parts required for each PEI type could potentially consist of thousands of NIINs with varying quantities for dozens of PEI types. Consolidating this information into an array will facilitate quick access as to what a PEI requires. Having access to this array via an index will facilitate efficient creation of each PEI. As model complexity increases, especially in regards to model constraints, so will the number of parameters.

5. DES State Space and Statistical Information

Derivation of important statistical information comes from the state variables that comprise the state space. The state space analyzed for this DES focuses on aspects of the parts inventory and how the PEIs move through the production process. Because each state variable is dynamic, we can derive statistical information about the system itself from the state trajectories produced by the simulation. This section explains the purpose of each state variable and important statistical information derived from it. In addition, the model can be adjusted to capture additional desired statistical data.

a. Arrival Queue Delay

The arrival queue delay state variable reflects how long PEIs are waiting before being disassembled. The time of delay is determined using the PEIs internal time clock. With this information the average delay time for PEIs awaiting disassembly can be derived. In the context of this thesis, this will indicate a deficiency in regards to available workspaces. If stock-out situations are severe enough to prevent PEI production goals from being met, new PEIs entering the system risk the chance of not having a workspace available. This metric can potentially provide planners with information to support increasing logistical capabilities if there is truly a high likelihood of increased stock-out situations. As a result, proactive decisions can be made in order to allow the production process to continue in regards to plant disassembly operations.

b. Arrival Queue

The arrival queue state variable maintains each PEI into the system in the actual order they arrive into the system. This state variable allows the program to determine the average number of PEIs in the arrival queue throughout the simulation. Having an idea of how many PEIs are not being accommodated provides additional insight to the problem. Some delay in the queue may be perfectly acceptable. However, a large number of PEIs being delayed even a short time may not be acceptable. This metric is particularly useful for planners when trying to determine exactly how plant operations should be modified in regards to either the acceptance of new PEIs or increasing logistical capabilities.

c. Assembly Queue Delay

The assembly queue delay state variable reflects how long PEIs are waiting before being reassembled. Again, the time of delay is determined using the PEIs internal time clock. As with the disassembly queue delay, the average delay time for PEIs awaiting assembly can be derived. In the context of this thesis, the only reason a PEI would not be assembled is if any of the required parts is not available (i.e., is in a stock-out status). The primary objective of CPIOM is to avoid the situation of PEIs not being produced due to stock-out situations. While it may not be realistic to say that all parts must be available in order to reassemble a PEI, it would certainly be conducive to the production process if this were the case.

d. Assembly Queue

Like the arrival queue, the assembly queue state variable maintains each PEI arriving into the assembly queue in the actual order they arrive. This allows the DES to determine the average number of PEIs in the assembly queue. This metric provides added insight as to the impact of stock-outs.

e. PEI Time in System

The PEI time in system state variable simply reflects the time it takes for each PEI to move through the entire production process, i.e., from arrival to completion. This allows the DES to determine the average time it takes for a PEI to be produced. While perhaps not as important a measure as the percentage of PEIs produced, this statistical measurement is a good measure of performance as to how efficiently PEIs are being produced. Comparing this metric to the overall time it takes to produce a PEI with no delays can provide useful information as to how adjustments to the system can be either detrimental or conducive to the process.

f. *PEIs Completed*

The PEIs completed state variable reflects the PEIs in a completed status. Once a PEI is reassembled it is placed into list of completed PEIs. The sole purpose of this state variable is for determining the percentage of PEIs completed at the end of the DES. This metric is considered the most important one of the system. The goal is to ultimately produce all of the PEIs by the end of the DES.

g. *Total PEIs Entering System*

The total PEIs entering system state variable reflects the number of PEIs entering into the system. The only purpose of this state variable in the context of this thesis is for calculating the percentage of production completed at the end of a simulation run as explained in the PEIs completed section. However, there are several other statistical calculations that may utilize this state variable.

h. *Number of Available Bays*

As the name implies, this state variable reflects the number of bays available at any given moment in simulation time. The number of available bays state variable also serves as the basis for determining if a PEI can be disassembled or not. The primary statistical measurement derived from this state variable is the average number of available bays available. Using this statistic, the average bay utilization can be calculated. This can be useful in determining how well production line resources are being utilized. A low utilization rate would indicate that the system resources are not being maximized. On the other hand, very high utilizations can result in unacceptably large delays. While this measure falls out of the scope of this thesis, further studies may reveal many interactions within the modeled process.

i. *Parts Inventory*

This state variable reflects the total inventory of the production process. This is the most dynamic state variable because it can potentially consist of thousands of parts and changes throughout the production process. Statistical measures can be derived at both the holistic inventory level or at the individual part level. The primary statistics derived from this variable is the average stock level quantities. This statistic will allow for other useful statistics to be determined such as overall total average safety stock cost and percentage of stock out that exists. The OMT in its current form does not allow for a determination of time the inventory or component thereof remains in a particular stock status i.e., stock out or safety stock status.

j. Part Order Quantity

The part order quantity state variable reflects the quantity of each part that needs to be reordered. As discussed above, the part order quantity is the decision variable output of CPIOM and will change each time CPIOM is run. The current program does not utilize this state variable for the purpose of statistical inference.

k. Deficient Parts Quantity

The deficient parts quantity state variable reflects the quantity of each part of a PEI that is unserviceable. As with the part order quantity state variable, this variable is not utilized for the purpose of statistical inference.

6. DES Event Graph

This section will briefly describe the event graphs associated with the DES. Detailed explanation of the event graphs is provided in subsequent sections. There are two event graphs used to outline our DES. Figure 12 reflects the event graph for the PEI arrival process and Figure 13 reflects the event graph for the PEI production process. The goal of the event graph is to represent the entire programming construct of the DES. Often, small elements of pseudo code exist within the event graph to represent the specific actions taken by a particular event. For the sake of visual clarity, only a few conditionals and variables are outlined in Figure 12 and Figure 13. The event graph can essentially be viewed as the DES's blueprint.

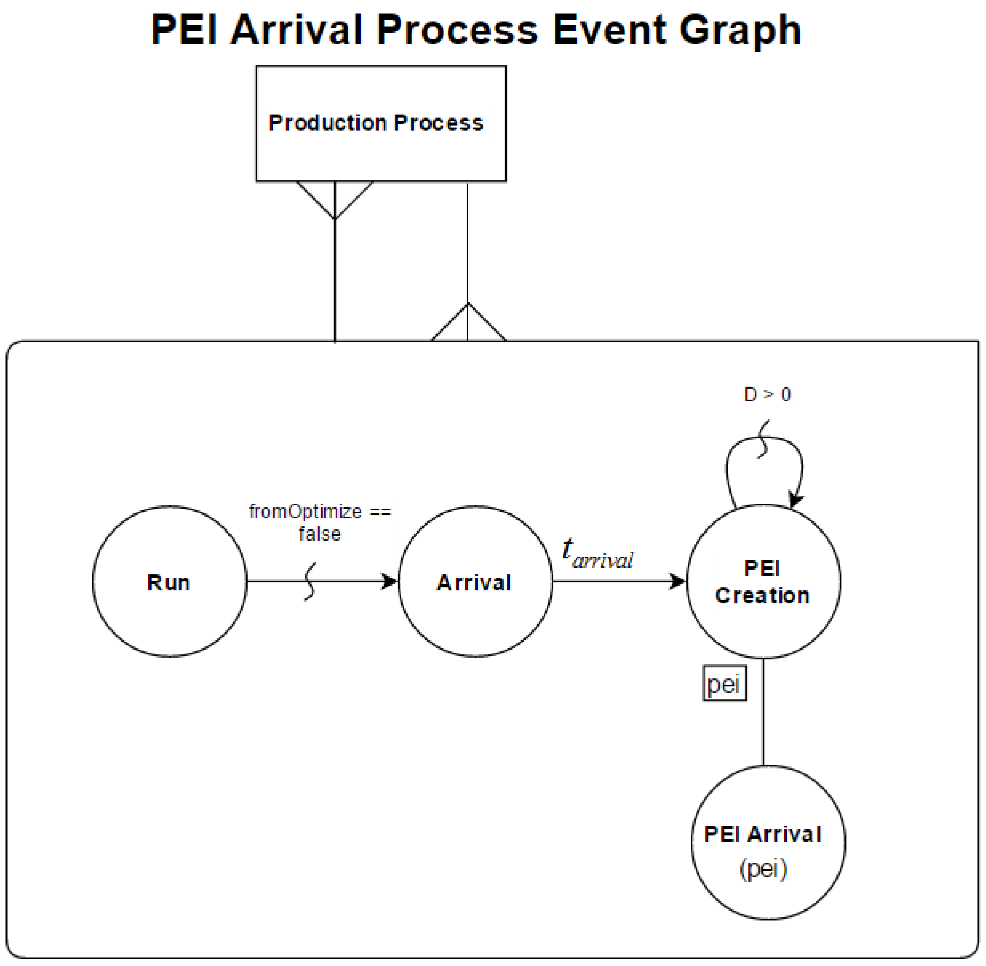


Figure 12. The PEI arrival process event graph programmatically outlines the process of generating a PEI entity.

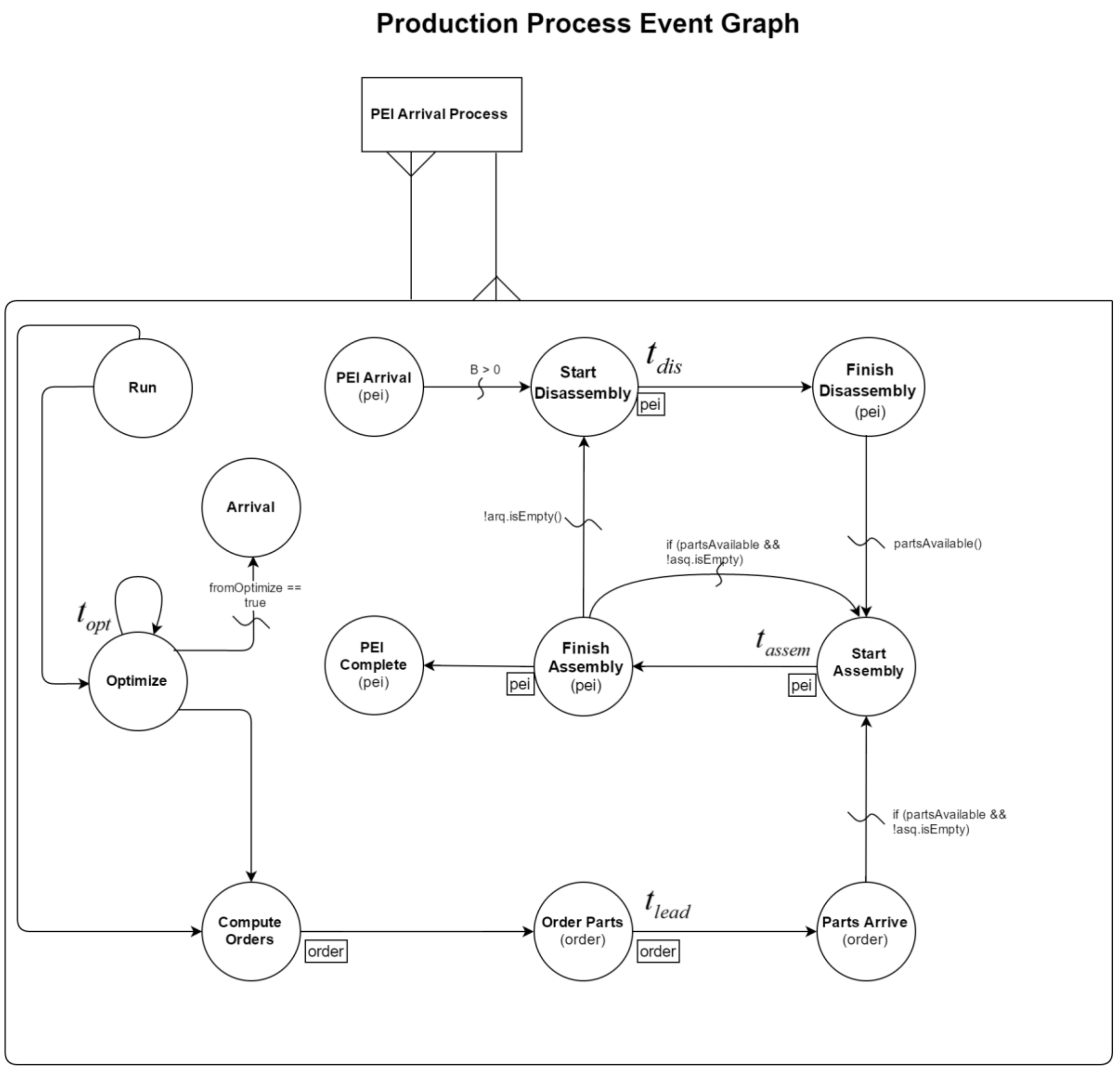


Figure 13. The PEI production process event graph programmatically outlines the production process for a PEI entity.

Notice that in both event graphs there are a series of lines leading from one process to the other. This means the two processes are "listening" to each other. Because some of the event list logic, the sequence of scheduled events do not necessarily correspond chronologically. In order to facilitate interpretation of the event graph, all state variables and parameters used within the event graphs are referenced in Tables 1 and 2. Table 1 provides a reference for all parameters and Table 2 provides a reference of all state variables. We will then discuss in detail each event associated with the event graph.

Table 1. List of parameters for the arrival process and production process event graphs.

Parameter	**Type**	**Initial Value**	**Abbreviation**
totalNumberBays	Int	User defined	K
NIINLeadTime	Double	varies	t_{lead}
PEIDisassemblyTime	Double	varies	t_{dis}
PEIAssemblyTime	Double	varies	t_{assem}
reoptimizeTime	Double	User defined	t_{opt}
arrivalTime	Double	User Defined	$t_{arrival}$
qtyPEIPerOptimize	Int	User defined	R

Table 2. List of state variables for the arrival process and production process event graphs.

State Variable	**Type**	**Initial Value**	**Abbreviation**
PEIComplete	ArrayList<PEI>	Clear	C
numberAvailableBays	Int	K	B
totalNumberAvailableBays	Int	0	T
arrivalQueue	PriorityQueue<PEI>	Empty	arq
assemblyQueue	PriorityQueue<PEI>	Empty	Asq
totalDelayInArrivalQueue	Double	NaN	DR
totalDelayInAssemblyQueue	Double	NaN	DS
delayInArrivalQueue	Double	NaN	Tdr
delayInAssemblyQueue	Double	NaN	Tds
totalTimeInSystem	Double	NaN	TS
timeInSystem	Double	NaN	Pts
inventory	Map<String,Integer>	original inventory	I
numberOfArrivals	Int	0	N
qtyPEIRemainingCreation	Int	qtyPEIPerOPtimize	D
fromOptimize	Boolean	False	O
peiServiced	Int	0	Sp
enterStartDisassemblyTally	Int	0	DT
partsArrived	Boolean	False	PA

7. PEI Arrival Process Event Graph

The PEI arrival process component within the DES is responsible for creating entities (i.e., creating new PEIs). We will explain in detail this component as outlined in Figure 12. This explanation is broken up by individual events. Reference Tables 1 and 2 for further clarification of variables used within the event graph. For visual clarity, Figure 12 is a generalized representation.

Programmatically speaking, the PEI arrival process is built on a simpler arrival process by creating a subclass of the Arrival Process. Instead of creating a new method or complicating an existing method, we add onto existing code without altering the original method. When a method of the same name is in the subclass, it is termed "extending" the method. Additionally, a subclass often adds methods and variables. In this case, the PEI arrival process "extends" the arrival process. This means the PEI arrival process will take on the functionality of the basic arrival process and add additional functionality.

a. *Run Event*

As its name implies, the run event is the first event that occurs within the DES. This event is executed once per simulation iteration and carries out the initial actions of the DES through via the reset() and doRun() methods [17]. The reset() method resets all applicable state variables to the starting state value. The state variables that apply to the PEI arrival process event include the number of arrivals, number of PEI arrivals remaining, and the from optimization variable. These variables are set according to Table 2. The doRun() method ensures variables needing to be tracked for statistical purposes are established. In this case, the numberOfArrivals variable is required for statistical tracking. At this point, the Arrival Event is immediately scheduled. Information is passed along when scheduling a new event to include the next scheduling time and relevant additional information. Additional information may include entities and variables. The information passed for the Arrival Event include the fromOptimization variable and the interarrivalTimeGenerator variable. The fromOptimization variable is not necessary in the simulations current state. However, it may be necessary if there are unique actions occurring on initial execution. The interarrivalTimeGenerator variable allows the simulation to randomly assign a time and can modified as the user sees fit.

b. *Arrival Event*

The Arrival event executes once the randomly assigned schedule time has been reached. The first action taken by this event is to keep track of the total number of PEIs entering the system. Each time a new PEI enters the system, the numberArrivals variable is increased by one for statistical purposes. At this point the simulation immediately moves onto the PEI creation event. Currently, the Arrival event schedules the next event for newly arriving PEIs in mass. The assumption is that the PEIs are staged and ready for processing.

c. PEI Creation Event

The primary purpose of this event is to facilitate PEI creation and update the master inventory. As mentioned earlier, this event is an extension of the arrival event. Because of this, the event is technically not an additional event but part of the arrival event. In fact, the first action of this class is to call the arrival event. However, we are treating it as a separate event rather than lumping it into the previously explained "Arrival" event for conceptualization purposes. Once the Arrival event completes its tasks, this event will instantiate a new PEI. We initially assume all parts are serviceable. As a result, the required parts for that PEI is added to the main inventory and the PEI itself. A determination of what parts are unserviceable will occur after the vehicle is disassembled. At this point, this event will immediately schedule the PEI arrival event passing along the PEI itself and numberArrivals variable. The final step is to reduce the numberArrivalsRemaining variable by one. This process will repeat itself until there are no further arrivals remaining. Only one PEI type will enter the system and all PEIs are scheduled to arrive at the same time with no variability in the number of arrivals. However, the PEI arrival process can be programmed to generate PEIs either deterministically or randomly for the both the quantity of PEIs and arrival frequency. How the PEIs will arrive is ultimately up to the user. For example, the user may choose 48 AAVs and 12 howitzers to arrive at the same time and an unlimited number of trucks to arrive randomly, depending on the scenario being modeled.

d. PEI Arrival Event

In relation to the PEI arrival process, no actions occur within this event. Because the PEI arrival process is independent, the production process must be paying attention to the PEIs that are arriving. Each time a PEI arrives, the PEI production process is listening for the PEI arrival event. Within the SimKit programming construct, this is known as an "event listener." Notice in Figure 12 and Figure 13 that there are a series of connector bars leading to a process. This means that each process is listening to each other.

8. PEI Production Process Event Graph

The DES PEI production process component is responsible for replicating the actual production process a PEI must go through. This process is outlined in Figure 13. As before, an explanation is broken into individual events. Reference Tables 1 and 2 for further clarification of variables used within the event graph. For visual clarity, Figure 13 is a generalized representation.

a. Run Event

This event follows the same methodology as explained for the PEI arrival process run event. The reset method sets all remaining state variables according to Table 2. Second, all csv files are instantiated for use by CPIOM. Third, the initial optimization occurs. Lastly, the main inventory is updated with the current safety-stock. Statistical tracking for all applicable variables is established via the doRun() method. This includes tracking changes in the assembly and arrival queues, number of available work bay, changes in the inventory, number of PEIs serviced, and overall changes within the system. The variables used vary widely depending on the objectives of your analysis. This event will also schedule an optimize event as dictated by the user. In addition, the compute orders event is scheduled with no delay.

b. PEI Arrival Event

The PEI Arrival event is "heard" from the PEIArrivalProcess and has five primary actions. First, it will establish the time that the PEI arrives to the production process. Second, every PEI will be placed in an arrival queue. In the case that the production line is unable to accept the PEI, the arrival queue will allow the simulation to hold the PEI(s) until the system can accommodate them. This event will also ensure any change in the arrival queue is tracked for statistical purposes. Finally, the PEI arrival event will check to see if there are enough bays available. If bays are available, the start disassembly event is scheduled. Otherwise, the PEI will remain in the arrival queue.

c. Start Disassembly Event

The Start Disassembly event has a total of four functions. The first function is to update the number of available bays as the result of a new PEI entering the production line. Change in the number of available bays is tracked. The second function is to remove the PEI from the arrival queue. The change occurring within the arrival queue is tracked. Third, the event will determine how long the PEI is in the arrival queue in order to track this statistic. Finally, the start disassembly will schedule the finish disassembly event to occur some to be determined time in the future. This will include passing the PEI to the next event.

d. Finish Disassembly Event

The Finish Disassembly event has many functions. Its primary function is to adjust the main inventory for each completed PEI. This is a two-phase process. First, the parts required for each PEI are added to the master parts inventory. Every PEI arrives with zero parts so no action is required on the part of the individual PEI inventory. Second, the condition of the PEI (i.e., unserviceable quantities of each NIIN) must be determined. The main inventory will be adjusted accordingly. The change in the new inventory is tracked. After the main inventory is adjusted, the PEI is added to the assembly queue. Change to the assembly queue state variable is tracked. At this point, the Finish Disassembly event will check to see if all parts are available as well as ensure PEIs still exist in the queue. If both requirements are met, the start assembly event is scheduled with no delay. Otherwise, the PEI will remain in the queue until the conditions are met.

e. Start Assembly Event

The Start Assembly event removes the PEI from the assembly queue and determines the PEIs assembly queue delay time. Changes in the assembly queue and PEI assembly queue delay time are tracked. The Start Assembly event then schedules the Finish Assembly event to occur when the PEI is reassembled. Scheduling the Finish

Assembly event (see below) requires the PEI to be passed along by the Start assembly event. It must then remove the parts required for the PEI to be reassembled. This will ensure the main inventory does not fall into a negative balance in the case where multiple PEIs are being processed at the same time. Finally, changes in the main inventory are tracked for statistical purposes.

f. Finish Assembly Event

The Finish Assembly event has six state transitions. Because the simulation is currently designed to add all parts at the same time, this event will restore the PEIs internal inventory. The bay being used by the PEI is then released for another PEI to use and the associated change to the bay is tracked. The total time the PEI has been in the system is recorded. The event then schedules the PEI complete event, passing the PEI. The event will then check to see if any PEIs are waiting in the arrival queue. If so, a Start Disassembly event is immediately scheduled so that awaiting PEIs can be processed. One issue that may arise for PEIs waiting to be processed in assembly queue is that there may be a large influx of parts arriving thereby facilitating more than one PEI being processed for assembly. In order to remedy this issue, the final action is to check and see if there are enough parts to accommodate additional PEIs being assembled. If additional PEIs are waiting in the assembly queue and sufficient parts are available, the Finish Assembly event schedules an additional Finish Assembly event.

g. PEI Complete Event

The PEI Complete event simply out-processes the PEI from the production process, and adds the PEI to the PEI completed list. This event can be modified for future iterations of the model to accommodate additional post production actions.

h. Optimize Event

The Optimize event’s primary action is to run CPIOM using the current inventory. In the context of this DES, the CPIOM must know the initial inventory. Prior to the DES start, the initial inventory is designated by the user provided input. Once the simulation begins the inventory is constantly in flux. As a result, the CPIOM must use the current main inventory of the process at the time of the Optimize event as the new initial inventory. It is important to clarify that the initial inventory is the safety stock that exists. Recall each PEI is essentially stripped of its parts. These parts are then placed in the main inventory. The safety stock is calculated by multiplying the total number of vehicles currently in the production process (i.e., uncompleted vehicles) by the parts required for each PEI and subtracting this from the main inventory. Any remaining parts reflect the safety stock and therefore the initial inventory to be used by CPIOM. Once the CPIOM has been run, the Optimize event will then schedule the next optimize event to occur according to the re-optimize parameter. In addition, the Optimize event will schedule the compute orders event and another arrival event.

i. Arrival Event

No actions occur in relation to the PEI production process. Recall that the PEI arrival process is listening for this event. When this event is scheduled, the PEI arrival process begins the process of bringing additional PEIs into the system.

j. Compute Orders Event

The primary purpose of the Compute Orders event is to ensure the accuracy of orders. One of the issues that can occur within the simulation is that previous orders may still be pending. Because the CPIOM does not consider pending orders, the compute orders event will screen all pending orders in order to ensure the same parts are not being double ordered. If there are instances in which the parts required to be ordered are actually less than what is currently on order, the orders will not be cancelled. This is assuming variations between the optimization model and the simulation will even out over time. In cases where the parts required exceeds the total quantity on order, only the difference between the quantity of parts on order and quantity of parts needing to be ordered would be put on order. Another feature of the compute orders event is that it will only process NIINs to be ordered if historical order data exists. In other words, the DES cannot process an order unless it can determine a lead time for the NIIN being ordered. Chapter IV will discuss the problem of having insufficient ordering data in more detail.

For each NIIN that requires an order, an order parts event is scheduled in which the NIIN and quantity to be ordered is passed along.

k. Order Parts Event

The Order Parts event simulates the actual "ordering" of parts. Critical to this function is determining the lead time of each NIIN to be ordered. This is accomplished through the use of historical order data and will be explained later on within this chapter. Once the lead time is determined, the parts arrival event is scheduled according to the assigned lead time. Again, the NIIN and quantity ordered is included when scheduling the parts arrival event.

l. Parts Arrival Event

The Parts Arrival event processes the parts when they arrive, updating the main inventory by adding the parts and tracking the change in inventory. The Parts Arrival event will then immediately notify the production line that new parts have arrived by scheduling a start assembly event.

D. PROGRAMMING IMPLEMENTATION

This section will provide a brief explanation of the various programming classes used to execute the simulation as well as critical components of the code. In addition, the source code will contain annotations throughout explaining in more detail the methodology used. The source code can be obtained by contacting Modeling Virtual Environments and Simulation (MOVES) at the Naval Postgraduate School.

1. Order Management Tool Data Entry

Data entry of required OMT input data is done via Microsoft Excel. For the current version of this OMT, the Excel workbook consists of five worksheets. The first sheet provides the NIINs being analyzed. For each NIIN, the 13 digit identification number, required quantity for each PEI, and initial quantity on hand is provided. The second sheet includes the condition history for a particular PEI. The sheet includes a listing of each PEI serial number and a tally of unserviceable parts for each NIIN. The third worksheet provides the order history. This includes the original order date and the arrival date for every NIIN ordered. The fourth worksheet provides the unit price of each NIIN. The fifth worksheet allows the user define the OMT parameters. This includes the safety-stock budget, number of PEIs entering the system, number of available work-bays, simulation time, optimization frequency, steady state time, and number of simulation runs. For the current OMT version, a new set of PEIs will enter the system each time an optimization occurs.

2. Simulation Time

Simulation time is in working days. One working day represents eight hours. While not important for the current system represented, the hours will be critical when modeling the system in more detail. Because the simulation does not stop running, one full quarter works out to be roughly sixty-six working days or 264 working days for a year. This calculation takes into consideration weekends and holidays. The program allows the user to input the simulation time for three primary variables namely total simulation time, re-optimization time, and steady state time. The re-optimization time establishes the frequency at which the simulation will run CPIOM. The steady state time is the time in which the user wants the simulation to start compiling statistical information. This allows the user to determine the steady state of the simulation, as explained in Chapter IV.

3. Java Classes

In order to more easily construct and make future modifications to the program, it is important to divide the program into classes. Classes are simply functional components or building blocks of a program. This “piece-meal” approach allows increased flexibility when developing future iterations of the program.

a. Input Data Processor Class

This class is responsible for processing all spreadsheet data for use by CPIOM and the DES. The class accomplishes this by systematically assigning relevant data to a series of reference databases called maps. The simulation references these maps for the duration of the simulation. Processing of the data must occur in a specific order. As a result, modifications to the data entry spreadsheet will result in errors. For example, the DES must reference the required parts first. Without knowing this basic information, the DES will not be able to carry out future logic.

b. Comma Separated Value (.csv) File Creator Class

CPIOM requires certain input files to be in comma separated value (csv) format, and this class is responsible for generating those files. There are eight .csv files required of CPIOM. With exception of the NIN_data.csv file, all of these files generate once prior to the simulation starting. The NIN_data.csv file must update information about the initial quantities of new parts each time prior to the program re-optimizing. This class initiates the required calculations for determining the initial-on hand inventory for each NIIN. These calculations include projecting required parts for PEIs pending disassembly while taking into account on hand new and used inventories as well as current parts orders. The files that need to be produced initially include the following:

NIN.csv is a listing of all the NIINs being ordered.

NIN_chance.csv is currently not used.

NIN_data.csv this contains important info concerning safety stock cost.

NIN_histogram.csv contains distribution of unserviceable parts.

V.csv lists the vehicle type(s).

V_data.csv provides number of vehicles.

V_NIN.csv provides the required parts per NIIN and PEI type.

c. Inventory Management Class

As the name implies, this class keeps track of new and used parts inventories.

d. Optimizer Class

This class is the critical link between the DES and GAMS. The Optimizer Class ensures all required directories exists and executes the GAMS program to run CPIOM.

The GAMS program correspondingly outputs the resulting files into the proper directory for later use by the DES.

e. Order Management Class

This class reads the recommended orders for each NIIN from the output file generated by CPIOM. The DES correspondingly places this data into a map for future use.

f. Lead-Time Calculator Class

This class works in conjunction with the data Input Data Processor Class. The purpose is to produce a map of histograms for lead times of each NIIN. The Lead-Time Selector Class explained next will use this map of histograms.

g. Lead-Time Selector Class

When ordering parts within the parts order event, the Lead-Time Selector Class will determine what the lead-time is. Lead-time is determined by randomly selecting from the data provided by each NIIN lead-time histogram.

h. PEI Class

The PEI Class defines the specific attributes of each individual entity entering the system. The DES in its current state contains a map attribute composed of the parts inventory. Simulation of a more detailed system model will require adding parts at different times. Having an inventory attribute will allow the DES to keep track of specific parts contained in an individual entity and thereby facilitating various events within the disassembly and assembly phases. While not currently used, each PEI also has an attribute reflecting how long it is sitting in a particular queue. Future iterations of the DES may also include using different PEIs. This will require an entity identifier attribute. Additional PEI variations within a single DES will inevitably result in creating additional classes focused on those specific PEI types.

i. Arrival Process Class

The Arrival Process Class tracks the number of entities entering the system and controls the flow and frequency of arriving entities.

j. PEI Arrival Process Class

PEI Arrival Process Class is an extension of the Arrival Process Class. This class executes the logic of the PEI Arrival Process Event Graph Figure 12. As each entity arrives to the system, this class will establish the attributes of each individual entity. The entities for this DES are the individual PEIs as defined by the PEI Class. This class will establish a PEIs individual parts inventory. A PEI arrives with all parts available whether they are serviceable or not. As a result, The PEI Arrival Process Class will assign each PEI a full inventory according to the user input.

k. PEI Production Process Class

This PEI Production Process Class contains the core logic of the DES. This class executes the logic of the Production Process Event Graph Figure 13.

l. PEI Condition Selector Class

Each individual PEI will arrive to the system in a certain condition. The PEI Condition Selector Class will determine the condition of each PEI. This occurs after the disassembly process for each PEI.

m. NIIN Availability Check Class

The sole purpose of the NIIN Availability Check Class is to ensure that parts are available in the master parts inventory. If a particular NIIN is not available, this class will prevent the PEI from entering the assembly process.

n. Simkit Chart Factory Class

This class creates the various histogram charts resulting from the DES. This is made possible using the open-source jfree library [4].

o. Histogram Class

The Histogram Class generates the various histograms within the simulation using the SimpleStatsTally class within SimKit's statistical library. The histograms enable the generation of the NIN_histogram.csv file and all output graphs.

p. Run PEI Production Process Class

This is the main class used to run the DES. The class initiates the required classes enabling the DES to execute. This includes creating an InputDataProcessor variable and ensuring only relevant data is available. Other variables instantiated include an InventoryManagement, PartsConsumption, and LeadTimeSelector variable. In addition, this class instantiates all required variables necessary for outputting statistical data into visual form via histogram charts. Lastly, the RunPEIProductionProcess Class tracks and initiates simulation repetitions.

4. OMT Statistical Tracking

SimKit allows for statistical information to be derived using built-in statistical libraries. Two functions derived from these libraries and used within the OMT include the simpleStatsTimeVarying and simpleStatsTally functions. The simpleStatsTimeVarying function is used when time is a dependent variable, as is the case in determining utilization rates. The simpleStatsTally function is used when time is not a dependent variable, as is the case when determining production rates. The main role of these functions are to keep a tally of each state transition as outlined in Chapter III. This is achieved using a SimKit method designed to pass state variable information about a state transition into the respective function. As a result, the functions keep a running tally of all state variables within the simulation allowing for statistical analysis.

IV. TESTING AND ANALYSIS

This chapter discusses the analysis of MDMC-provided data, the process used to verify the OMT, and finally demonstrates the basic OMT statistical utility. It will also provide the methodology for program verification and key findings resulting from the verification process. The demonstration portion of this chapter will provide a small sampling of the DES program utility in regards to statistical output.

A. DATA ANALYSIS AND VALIDATION

An analysis of MDMC data was conducted prior to building the OMT program. This analysis provided important insights because it helped to define how the program processes the data. From this analysis came two key findings. First, individual NIINs cannot be considered as independent variables. Second, the order history data provided by MDMC proved to be statistically insignificant for the parts requiring analysis.

1. Unserviceable Parts Distribution

As mentioned earlier, a key finding found when researching the distribution of unserviceable parts is that parts cannot be treated independently. Using historical data to look at the distribution of unserviceable parts for six NIINs associated with the Amphibious Assault Vehicle (AAV), it is determined that for certain parts a dependent relationship exists [1]. For example, the probability of an AAV requiring NIIN 015421278 (torsion bar type 1 of 4) using a binomial distribution is 17.7 percent. As shown in Figure 14, the probability distribution of this NIIN for 12 vehicles (36 parts) can be approximated by a normal distribution. This assumes independent failures. However, when plotting the data for 96 vehicles a very different distribution is displayed, as seen in Figure 15. In fact, Figure 15 shows this relationship occurs for other torsion bars as well. This implies that part failures cannot be assumed independent of each other (both failures for a given part and failures for two different parts). For two AAVs, Figure 16 shows the difference between an independent relationship and a dependent relationship for NIIN 015421278. Two dependencies exist. First, for a given part there are multiple failures. CIPOM considers this using the convolution technique explained in

Appendix A. The second dependency is for multiple parts (among parts). CIPOM ignores this situation, as it is too difficult to convolve within the optimization. The OMT uses a bootstrapping technique in order to determine the proper probability distribution for part failures [26]. This is done by randomly selecting a PEI from historical data.

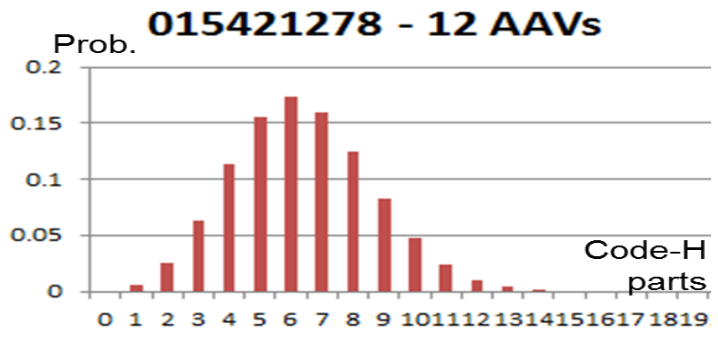


Figure 14. The probability distribution of NIIN 015421278 for 12 vehicles (36 parts) is approximately normal when using a binomial distribution.

Source: [1] J. Salmeron, A. Buss, T. Curling and M. Kress, "Plant utilization at Marine Corps Logistics Command," Naval Postgraduate School, Monterey, 2014.

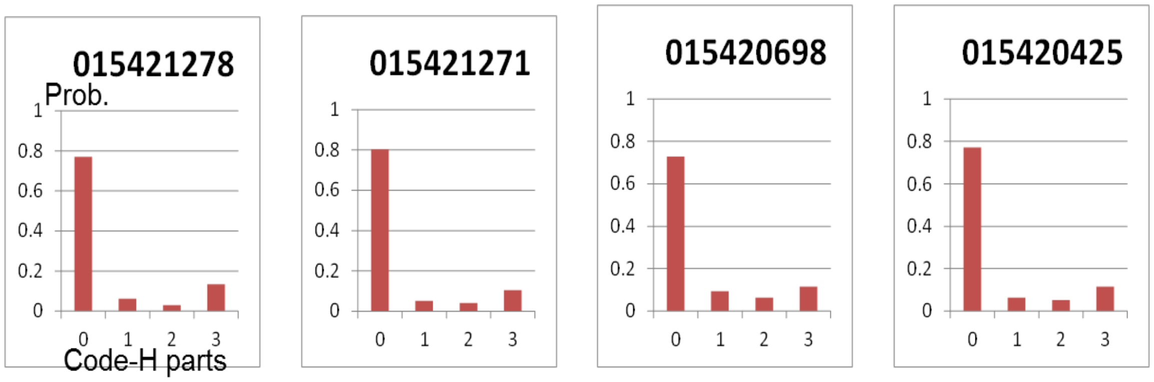


Figure 15. Distribution based on actual data for four different torsion bars. Histograms indicate part failures for torsion bars are not independent. If one torsion bar is in serviceable condition, then all torsion bars are likely serviceable. The converse is true for an unserviceable torsion bar.

Source: [1] J. Salmeron, A. Buss, T. Curling and M. Kress, "Plant utilization at Marine Corps Logistics Command," Naval Postgraduate School, Monterey, 2014.

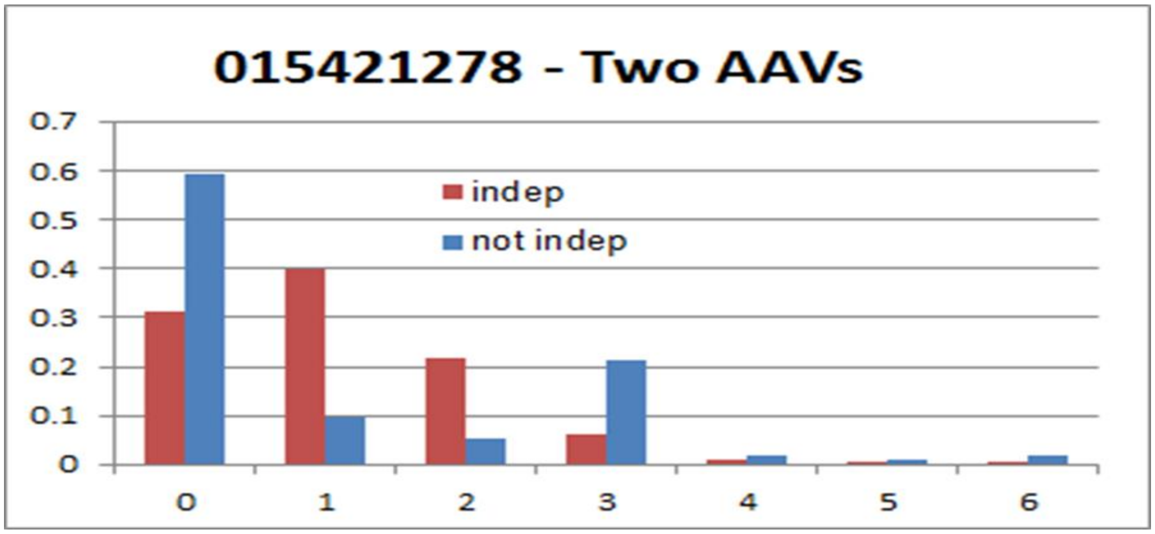


Figure 16. Graph shows the difference between an independent relationship and a dependent relationship for NIIN 015421278.

Source: [1] J. Salmeron, A. Buss, T. Curling and M. Kress, "Plant utilization at Marine Corps Logistics Command," Naval Postgraduate School, Monterey, 2014.

2. Order History Analysis

Order history analysis of selected NIINs found MDMC provided order history data to be statistically insignificant. The provided order history data includes 178,198 orders going back two years. Of the 18 NIINs requiring analysis, only nine NIINs had sufficient associated data in order to allow for statistically significant estimation of arrivals. Associated data includes NIIN price, deficiency for NIIN, and at least one order per NIIN. For this analysis, a Java program was created to filter parts order data and output as histograms. The histograms show for each NIIN the number of order occurrences by lead-time. Of the nine NIINs having sufficient associated data, the maximum sampling size that occurred is 27, as shown in Figure 17. The figure shows that over 65 percent of parts ordered appear to have been immediately available when ordered. However, orders took from 120 to 600 days to arrive when the parts were not available. Without a larger sampling size and normal distribution of parts order history, these data are statistically insignificant. An investigation of the validity of the MDMC provided data was not conducted. It is possible that parts were procured outside of the system through various methods. Because of this finding, the primary objective of this thesis is to demonstrate a proof of concept. For MDMC to use the OMT for future analytical research, it is critical to use accurate and statistically significant data. Nonetheless, important insights as to how the OMT can assist MDMC will be explained in future sections.

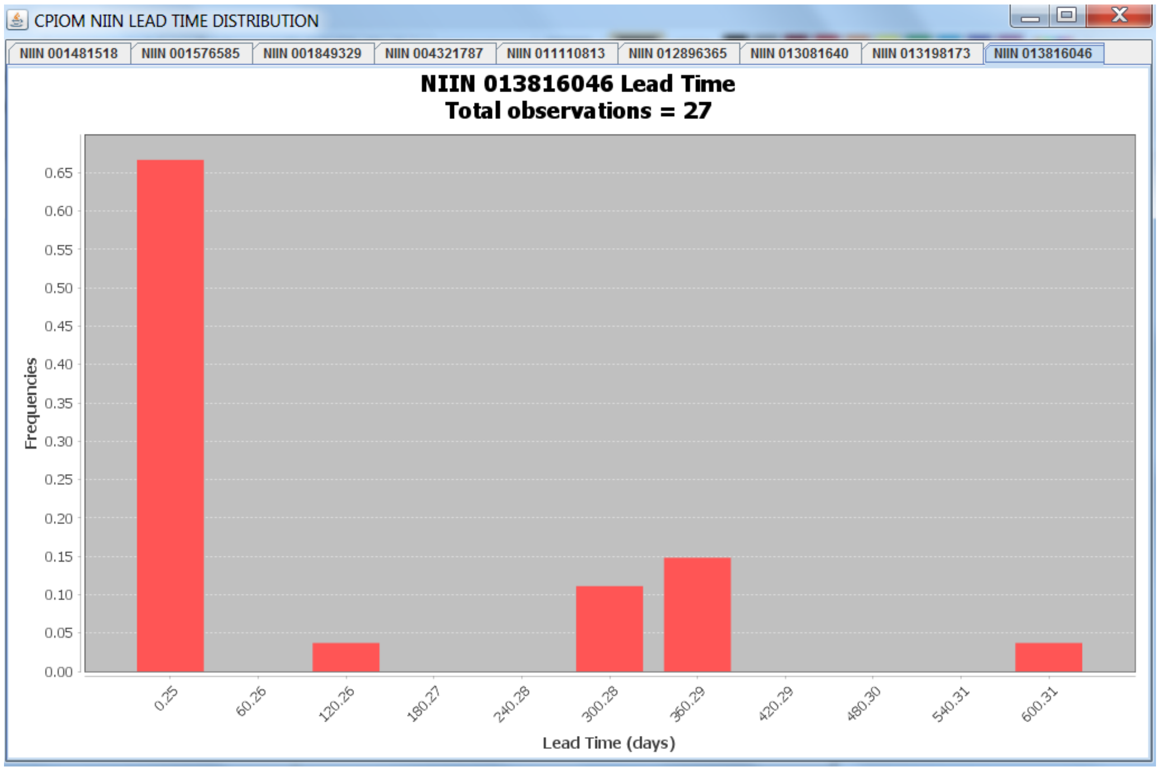


Figure 17. Order analysis from MDMC provided data. This figure provides a snapshot of the order history for a required NIIN.

B. PROGRAM VERIFICATION

The goal of the verification process is to ensure the DES program functions as programmed. As discussed earlier, this is a critical step in ensuring the simulation and all of its components can be successfully validated. If the program is not verified functional, it cannot be validated against the real world process. It is important to note that the functionality of the simulation was continuously verified throughout the development process. By the time the final component of the program is developed, there were very few to zero unknown discrepancies. Only unresolved discrepancies will be mentioned, all of which had to do with the data rather than the model's functionality. These discrepancies as well as the key findings affecting the model development will be explained over the next section. This will include discussing unserviceable parts distribution, unserviceable parts distribution csv file, negative part balances, and large parts requirements.

1. Unserviceable Parts Distribution CSV File

The OMT processes NIINs in the order received when generating the NIN_histogram.csv file. While negligible, the order of NIINs within the.csv file does matter. This is indicated when running CPIOM with NIINs placed in different orders within the .csv file. An explanation for why this exists is unknown, but it is nonetheless a consideration to take into account for future work. This is only relevant to CPIOM.

2. CPIOM Failures for Large NIIN Orders and PEI Quantities

Exceeding established parameters within the CPIOM will result in a failed solution and subsequent DES failure. If a failure occurs, review the input data and adjust the CPIOM as necessary. During testing, two failures occurred when running the OMT. However, failures for other reasons may occur as input data sets expand. The two failures encountered during our testing were the result of parameters being exceeded for the maximum number of vehicles (*nK)* and total number of possible orders of a given NIIN *(H)* in any run. Increases to the *nK* and *H* values allowed for successful calculation.

3. Elimination of Analyzed NIIN for Insufficient Data

For NIINs requiring analysis, the OMT will automatically eliminate NIINs for which insufficient data are available. This will occur when the NIIN does not have an associated order history, condition history, unit price, or required quantity for a PEI. If the NIINs requiring analysis are missing within the optimization output, check to see if information is missing for that part.

4. Future Negative Parts Balances

When running the simulation, there are times when the initial parts balance is negative. This is normal. On occasion, CPIOM's recommended quantity of parts ordered falls short of expectations. This has to do with the probabilistic nature of the modeled system. For example, there are situations where consumption of a particular NIIN is abnormally high for the order quantity. This results in initial quantities being negative. Recall that the CSV File Creator Class is responsible for calculating the correct initial parts balance.

C. CPIOM AND DES INTERFACE VALIDATION

Using the original baseline test case discussed in chapter two, the DES program verified that the CPIOM was interfacing properly with the DES. This was accomplished by stopping the simulation in stream and running the CPIOM independently using the same input variables used within the DES program, namely the current inventory. If the independent CPIOM run matched up with the DES program CPIOM run, the interface was confirmed to be working properly.

D. SCENARIO

With the OMT verified, an arbitrary sample scenario was used to demonstrate the utility of the OMT. Key input variables for this scenario are found in Table 3. The scenario is designed to handle only one vehicle type. The vehicle type for this scenario is the AAV. This scenario uses a triangle distribution for assembly and disassembly times centered on 3 weeks and 7 weeks respectively for a total mean processing time of 10 weeks per vehicle. This equates to a total mean value of fifty work days assuming there are no modifications to the production schedule. A total of six NIINs will be used in this scenario. NIIN lead time distributions are generic and range from 1 day to 6 months. The lead times are derived as discussed in the Lead Time Selector Class section. The conditions of individual vehicles are generated using fiscal year 2012 and 2013 data for a total of 96 AAV. The scenario is set up to process a total of 12 AAVs per quarter for one full year. Each time new vehicles arrive into the system, parts will be ordered according to the CPIOM output. The vehicles will arrive in bulk (i.e., there will be no delay from one vehicle arriving to the next). The simulation will conclude at the end of the simulation year. The production plant is capable of processing 12 vehicles at a time. There are no limitations in regards to available employees at this time.

Table 3. This table reflects the key scenario input variables used for the OMT demonstration.

Variable	**Value**
Optimization Period	3 months
Safety-Stock Budget	$50,000
Simulation Time Period	1 year
Total NIIN types	6
Total AAVs Per Optimization Period	12
Total Work-Bays Available	12
Assembly Time Per AAV	35 work days (ranges from 30–42 days)
Disassembly Time Per AAV	15 work days (ranges from 13–17 days)

E. STATISTICAL OUTPUT

The primary focus of this particular simulation run is aimed at determining the overall effect of the CPIOM output on plant production effectiveness and efficiency. There are a total of five specific metrics being produced by this demonstration. The first four are measures of performance showing how well the plant utilizes its resources and where inefficiencies may exist. This includes the average delay a PEI incurs prior to being disassembled and reassembled, the total average time a PEI is in the system, and the average utilization rate of the plants assembly bays. The fifth metric, production rate, is the bottom line of how well the plant achieves its production mission. For this demonstration, the program is set up to run the one-year scenario a total of one thousand repetitions.

1. Simulation Steady State

Prior to running a scenario for analysis, it is important to establish the simulation's steady state through statistical analysis. Since the system being modeled does not start from scratch, the simulation needs to "warm up" in order to remove any bias associated with the (somewhat arbitrary) initial state. As in the real world, there is the possibility of previously existing PEIs and NIIN orders existing at the beginning of any given year. In order to determine a steady state, the simulation will run multiple iterations until reaching a steady state. For this scenario, the first iteration will involve running the scenario previously explained for one year. The second iteration will run the simulation for a total period of two years with only the second year’s statistical data recorded. The third iteration will run the scenario for a period of three years with only the third year’s statistical data recorded. Subsequent iterations will follow this pattern until a steady state is achieved. Determination of when the steady state is achieved is accomplished by comparing the statistical data for the production rate variable from one iteration to the next. If no significant difference exists, then the assumption is that a steady state has been achieved. The results of the iteration achieving a steady state are used for further examination of the system being analyzed. Figure 18 shows the comparative analysis of five simulation iterations. The steady state is not achieved until the fourth year of simulation.

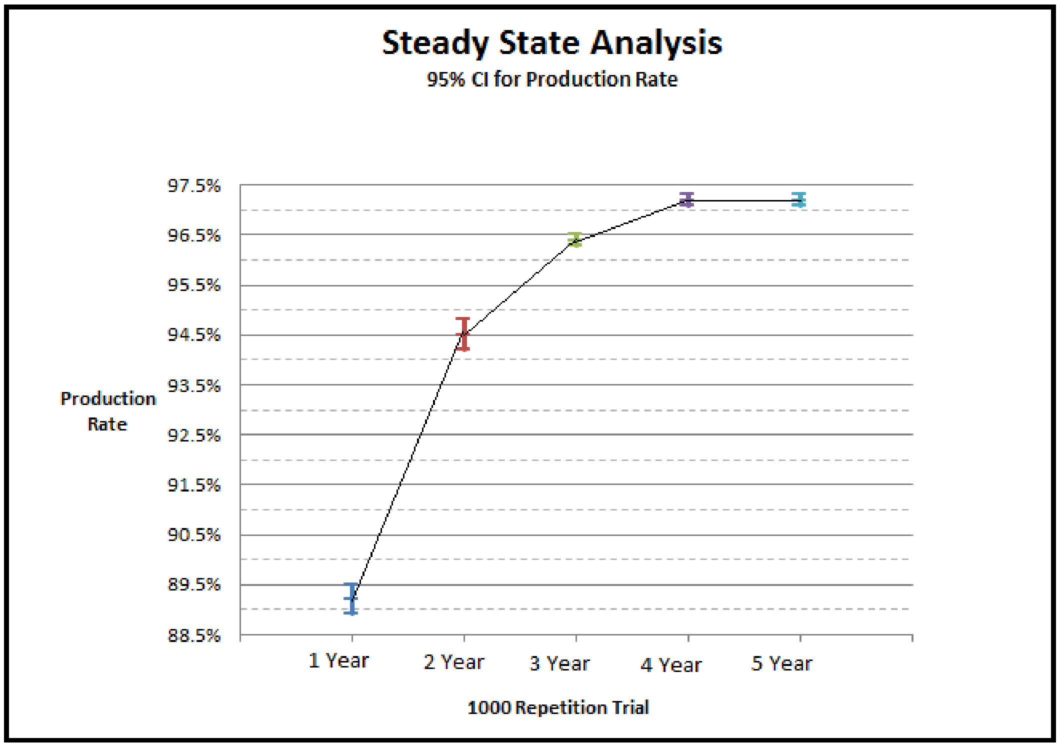


Figure 18. Steady state analysis for the scenario required the simulation to run for four years before reaching a steady state.

2. Average PEI Delay in Arrival Queue

All PEIs will arrive in the arrival queue at the same time. If a work bay is available, the PEI will immediately fill the bay and no delay in the arrival queue is incurred. Figure 19 reflects a relatively small delay in the arrival queue, averaging about seven workdays for 13.5 percent of the simulation runs. In rare instances, delays of up to 27 workdays occur. However, over 50 percent of simulation runs show delays ranging between three to eleven workdays. One important note to make about this metric is that it does not factor in the delay incurred upon the simulation ending. The delay is only calculated at the time the PEI leaves the queue. This also applies to the assembly queue delay metric explained next. As a result, the actual delay is possibly slightly higher than what is actually being captured by the DES. Nonetheless, the general idea of what may happen can be deduced.

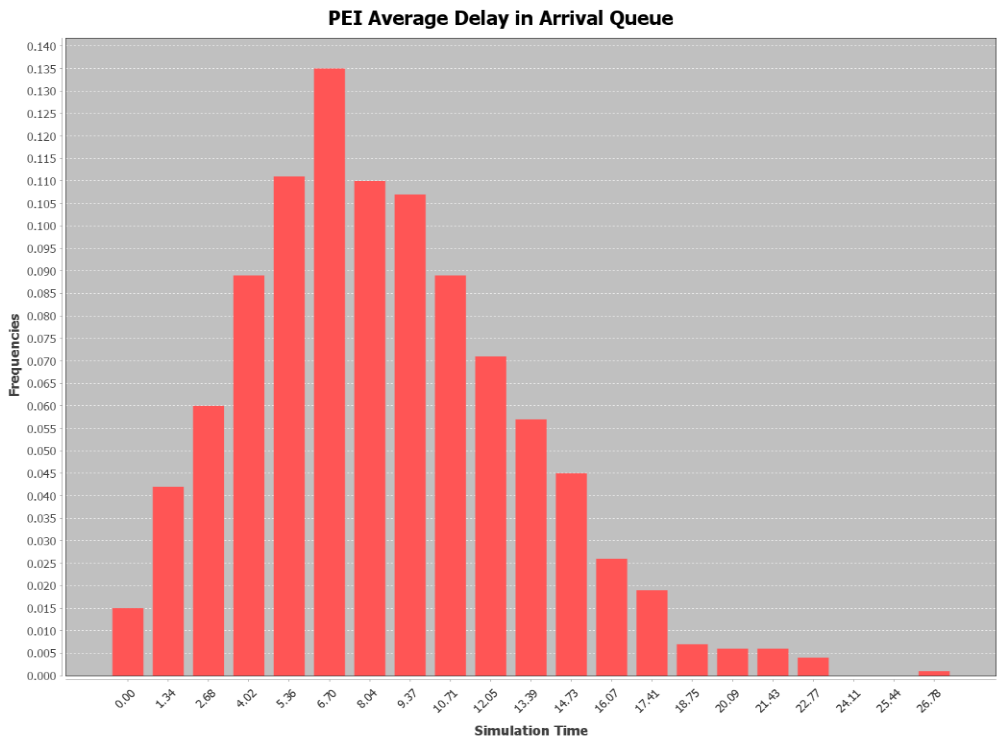


Figure 19. Simulation results reflect a relatively small PEI delay in the arrival queue. The average delay is about seven work days for 13.5 percent of the simulation runs. In rare instances, delays of up to 27 workdays occur.

3. Average PEI Delay in Assembly Queue

Once the PEI has been completely disassembled, the PEI is placed into an assembly queue until it can be reassembled. If parts are available to reassemble the PEI, it will immediately enter the assembly process and no delay in the assembly queue is incurred. Figure 20 shows about 45 percent of simulation runs having an average delay within the assembly queue ranging from seven to ten workdays. The delay existing within the assembly queue is purely the result of parts not being available. In instances where severe parts deficiencies exist, this will most likely have an effect on arrival queue delay. Seeing that there are sufficient bays available and the frequency of arriving parts is not severely deficient, delays existing within the arrival queue exist due to a backlog in the assembly queue. Because of the optimization component within the DES, the simulation is able to optimally adjust to the stochastic nature of arriving parts. Again, the delay represented in Figures 19 and 20 is most likely higher due to the assembly queue delay of existing PEIs not being captured at the end of the simulation run. In other words, this is a conservative representation of delay incurred.

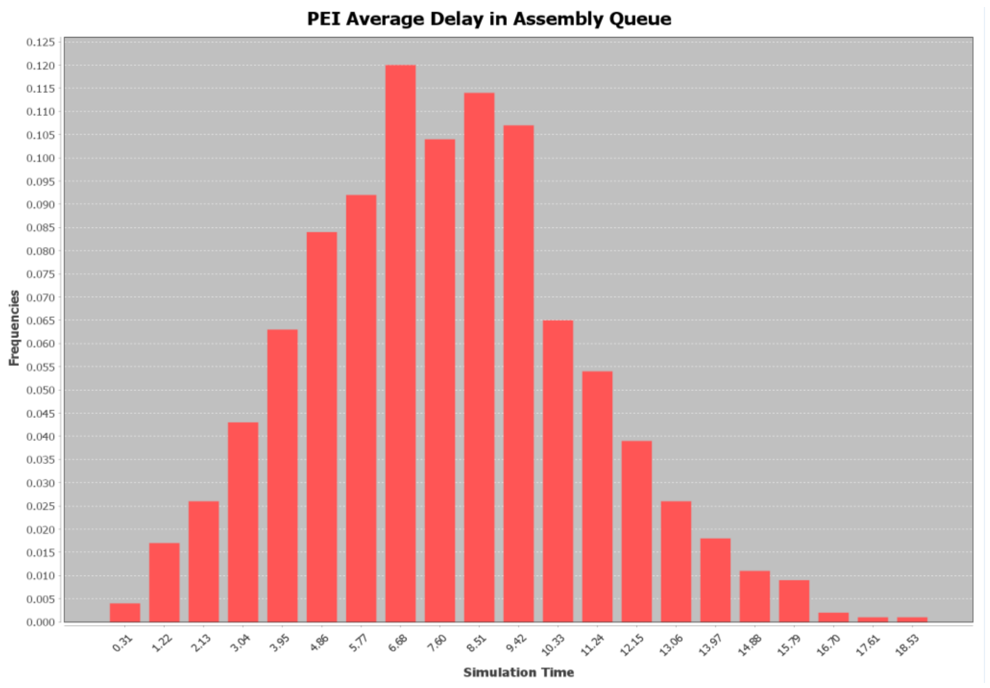


Figure 20. PEI average delay within the assembly Queue. The simulation results in about 45 percent of simulation runs having an average delay ranging from seven to ten workdays.

4. Average PEI Time in System

The overarching metric covering delay of production time is captured by the average PEI time in system metric. This statistic is particularly useful when looking at the system from a holistic standpoint. While inefficiencies may exist within certain locations of the process, they may be acceptable from a holistic perspective. There may be situations where it may not be feasible to reduce the delay in a certain location. It shows the combined effect of delay and production time. Figure 21 reflects no observations of PEIs incurring zero delays in production time. The majority of runs reflect average PEI total time in system between 61 to 72 days. In this scenario, managers can expect overall production to be delayed by 12 to 22 days in 57 percent of simulation runs. This would be an important point to consider when evaluating the utilization of human resources. The current simulation model only considers work bays. Future simulations may include adding employees as a state variable in order to track worker downtime. An example of this is seen in the next section when discussing production line utilization rates.

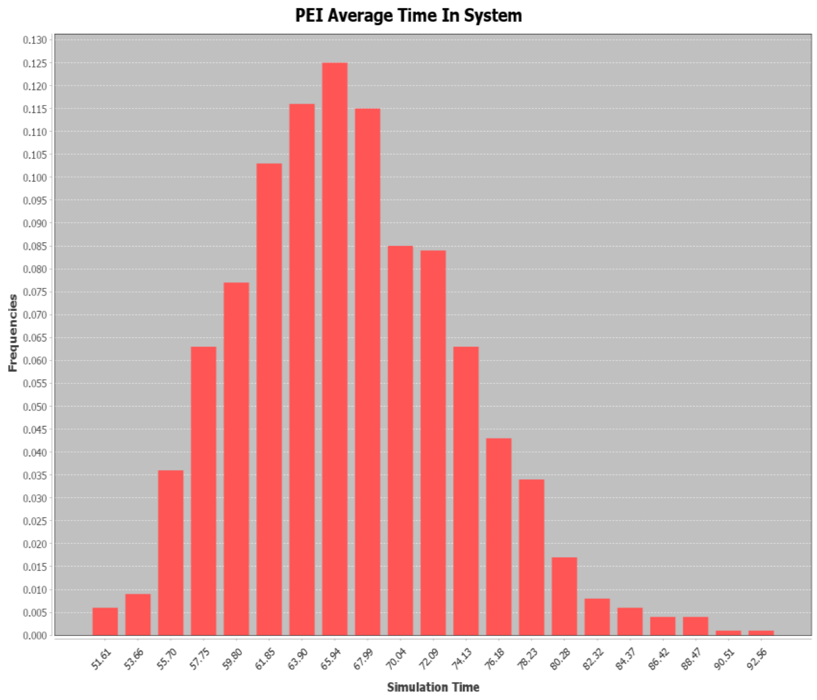


Figure 21. Average total time of a PEI in the system. Managers can expect overall production to be delayed by 12 to 22 days in 57 percent of simulation runs. Within the scenario, it takes 50 work days to produce a PEI on time.

5. PEI Production Line Average Utilization Rate

Showing the utilization of production resources may be a useful indicator of where inefficiencies may exist within the process. If even a small deficiency exists, it may allow for critical adjustment to the production process in order to cover shortfalls in other areas. For this simulation, the average utilization rate for the PEI production line reflects the percentage of bays being used throughout the simulation. From Figure 22, the average utilization rate for the production line is 100 percent during 14 percent of all simulation runs. 100 percent of the runs have a bay utilization rate greater than 90 percent. The situations where lower utilization rates exist within this DES construct is the result of too many bays existing in the first place and/or PEIs completed ahead of schedule. Because of these two situations, bay vacancies are created thereby reducing the utilization rate. When analyzing the utilization rate, it is important to take overall delay within the system into consideration. The combination of high utilization rates and low delay rates indicate the system is running optimally. Assuming few parts order delays, a high utilization rate with a high delay rate may point to inefficiencies within the system. In this case, we see utilization rates between 90 to 100 percent and relatively low delay rates of 12–22 workdays. These metrics can be compared to established parameters for determining estimated system efficiency and effectiveness, if such exist.

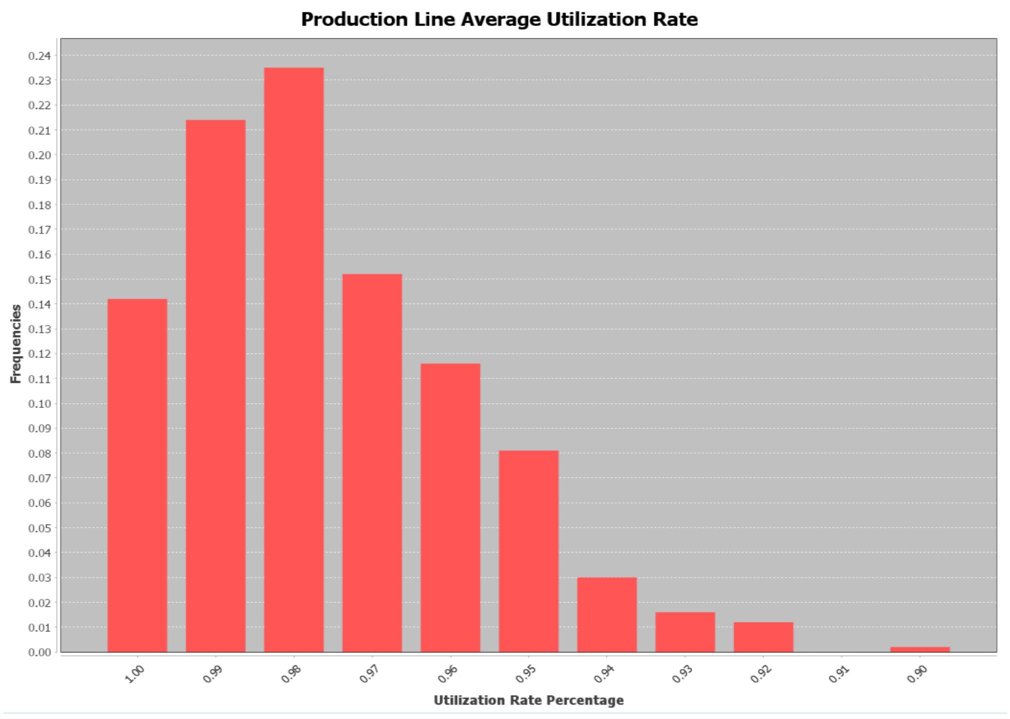


Figure 22. Work-bay utilization rate. During 14 percent of all simulation runs, the average utilization rate for the production line is 100 percent. All runs have a bay utilization rate greater 99 percent.

6. PEI Production Rate

The production rate is viewed as the primary measure of effectiveness. This metric is the percentage of total PEIs produced for PEIs entering the production process at the end of each simulation run. While the optimization model will provide the best combination of NIINs and associated quantities to order, it will not portray the variability in production output as a simulation can. From Figure 19, the simulation reflects over 90 percent of runs will fall short of the full production rate even with optimal order quantities being used. As discussed within the steady state analysis, the mean production rate achieved over one thousand simulation runs is 97.2 percent.

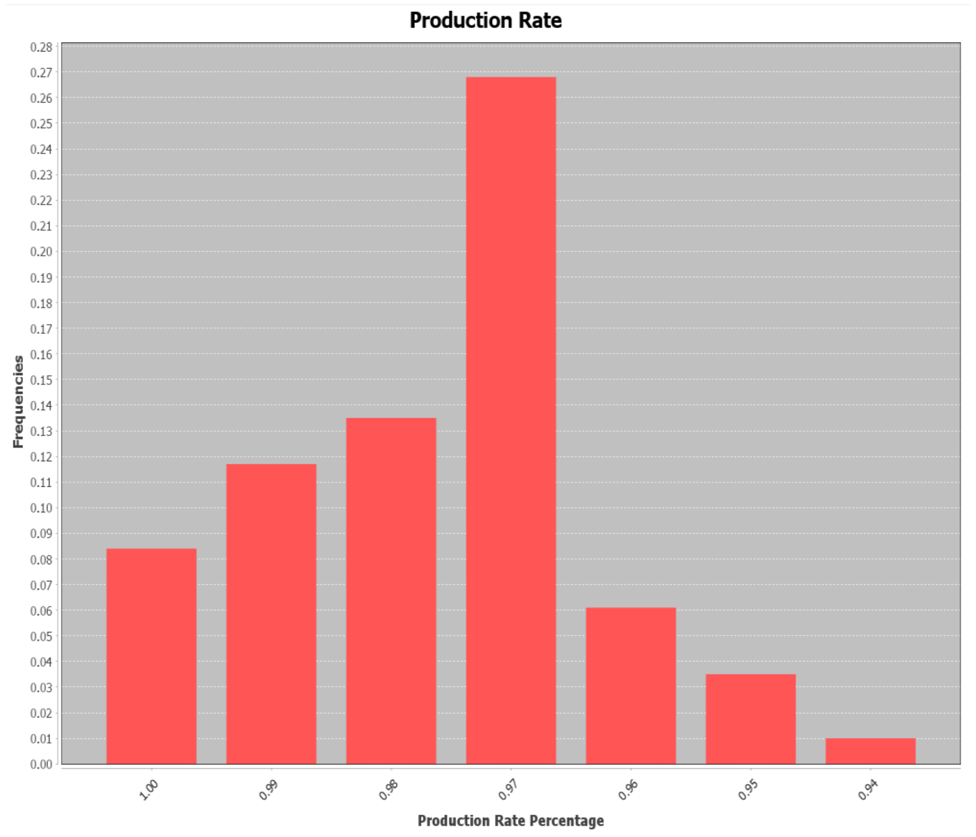


Figure 23. Histogram for production rate. The simulation reflects over 90 percent of runs will fall short of the full production rate.

7. Other Metrics

Several state variables could potentially be analyzed depending on the objectives of the organization. Employee utilization is one example that we mentioned briefly. Other variables include inventory levels, stock out costs, safety stock costs, and overall expenditure costs. The complexity of the system being modeled and simulated increases the complexity of potential analysis. It is ultimately up to the organization to identify the degree of complexity to model in the system and associated metrics.

V. CONCLUSIONS, RECOMMENDATIONS, AND FUTURE WORK

Several key insights and observations can be derived from the work provided by this thesis. In regards to the primary purpose of demonstrating to MDMC the utility of combining optimization and DES, there are two main observations. First, data input validation is, not surprisingly, critical to the effective use of any analysis tool that utilizes optimization and simulation techniques. Without quality historical data to fuel CPIOM and the DES, the outputs of the simulation will result in an inaccurate representation of the system. The second observation is that a joint optimization and DES construct provides valuable information when analyzing a complex stochastic system. In general, the use of optimization modeling is fundamental when dealing with systems involving many decision variables. The production lines at MDMC involve thousands of variables and thereby make them an ideal candidate for applying optimization. Because of the wide range of complexity and variability existing within the MDMC, DES is an essential element of the optimization process.

Throughout the development of the OMT program, a building approach has been used, which allows the program to be easily modified in the future. In particular, CPIOM and the DES are completely independent within the coding structure. The only element connecting the two components are the data inputs and outputs. This is important because it allows developers to make changes in each components code independently. Of course, if changes result in modifications to output or input structure, this would result in a corresponding coding adjustment. The building block approach to the design of the DES allows the developer to increase model complexity incrementally. This is important since the organization will want to have more than just a single simple component of their system. This is especially true when parts and employees can be cross-decked among multiple platforms. For example, if a particular part is utilized by both a tank and a truck or a painter is required for all PEI types.

This thesis has demonstrated the OMTs practical utility. This includes providing a user with output metrics for delays within the system, utilization rate for work-bays, and overall system production rates. The primary OMT output for the scenario in this thesis is that over 90 percent of simulation runs will fall short of the full production rate. Additional analysis reveals total delays of 12 to 22 days in 57 percent of simulation runs and at least a 90 percent work-bay utilization rate among all runs. Using information such as this provides insights into system efficiency and effectiveness. This in turn can be used to develop organizational policy. Keep in mind that this demonstration uses a fictional data set for NIIN order history and simulates a basic system. Future refinements and complexities can be added depending on organizational analytical and mission objectives.

As the program is developed and refined further, the addition of new components inevitably results in the ability to analyze additional variables. The DES program is currently programmed to analyze only five output variables of the system: average PEI delay in arrival queue, average PEI delay in assembly queue, average PEI time in system, average work bay utilization rate, and average production rate. Of course, many more variables can be analyzed depending on the model's complexity. For MDMC in particular, financial accounting is an important consideration not currently analyzed within the simulation. These financial variables include overall safety stock costs, stock out costs, and budget. These same variables can be further broken down to the costs associated with individual NIINs. Other variables to consider would be employee utilization (if employees were explicitly added to the model), inventory stock levels, and PEI production rates by PEI type. In summary, the effective combined use of DES and optimization modeling provides a potentially powerful resource for analyzing many variables within a complex system.

APPENDIX. CPIOM MATHEMATICAL FORMULATION

The below information is from [1]. This appendix describes the formulation of the CPIOM model. After the formulation, the functionality of each constraint involved is described.

A. INPUT DATA: INDICES, INDEX SETS AND PARAMETERS

$i \in I$, critical parts, also known as NINs

$v \in V$, vehicle types

$v_i \in V$, vehicle type that has part i

n_v^V, number of vehicles type v

n_i^I, number of parts i in each v_i vehicle

n_i, total number of parts i. Calculated as $\sum_{v \in V_i} n_v^V \, n_i^I$

b, budget for safety stock level

c_i^{SO}, cost of each stockout of part i

c_i^{SS}, cost of each part i in safety stock (unused inventory)

q_i^0, initial stock of part i

δ^{q^0}, one if the initial stock (q^0 vector) counts against safety stock budget, and zero otherwise

1. Approximation of probability distribution for replaceable parts

Method 1 (independent parts): Assumes all part replacements are independent, even for parts of the same type within the same vehicle. E.g., “a broken torsion bar in vehicle 1 does not affect the probability that the next torsion bar on that vehicle is also broken.” The formulation required is as follows:

p_i , probability that each part i needs to be replaced. This probability is estimated based on historical data as follows:

total parts of type *i* replaced / total parts of type *i*

D_i , random variable for demand (# of parts) i that need to be replaced. $D_i \sim \text{Binomial}(n_i, p_i)$

$k \in K_i$, index for probability levels for part *i*, $K_i = \{0,1,2,...,n_i\}$ (see below)

d_{ik}, p_{ik} , demand for level *k*, and probability for that level, for item *i*:

$$d_{ik} = k, \forall k \in K_i$$

$$p_{ik} = \Pr\{D_i = d_{ik}\} = \binom{n_i}{k} p_i^k (1 - p_i^k)^{n_i - k}, \forall k \in K_i$$

Remark: We are modeling $d_{ik} = k, \forall k \in K_i$ because the number of parts is small and so $D_i \sim \text{Binomial}(n_i, p_i)$. However, if the number of parts were too large we would have to group parts into other levels.

Method 2 (partial dependence): Assumes part replacements for different part types within the same vehicle or for the same part type in different vehicles are independent. However, part replacements of the same type within a given vehicle are not independent. E.g., “a broken torsion bar in vehicle ‘A’ does not affect the probability that the vision block on vehicle ‘A’ is also broken; but, it does affect the probability that another torsion bar on vehicle ‘A’ is broken.” The formulation required is as follows:

p_{in} , probability that n parts of type i need to be replaced, for $n = 1,...,n_i^I$. This probability is estimated based on historical data as follows:

total vehicles requiring *n* parts of type *i* replaced / total number of vehicles

D_i^V , random variable for demand (# of parts) i that need to be replaced in a vehicle of type v_i. That is:

$$p_{in}^V = \Pr\{D_i^V = n\} = \frac{\#\text{historical vehicles with } n \text{ replacements of } i}{\text{number of historical vehicles}}, \ \forall n = 1,...,n_i^I$$

D_i , random variable for demand (# of parts) i that need to be replaced. $D_i \sim \sum_{m=1...n_{v_i}^V} D_i^V$

The above convolution (sum of i.i.d. random variables) can be calculated using the below notation and recursive procedure:

$k \in K_i$, index for probability levels for part *i*, $K_i = \{0,1,2,...,n_i\}$ (see below)

d_{ik}, p_{ik}, demand for level *k*, and probability for that level, for item *i*:

$$d_{ik} = k, \forall k \in K_i$$

1. Calculate $D_i^{V2} = D_i^V + D_i^V$:

$$p_{ik}^{V2} = \Pr\{D_i^{V2} = k\} = \sum_{j=0,\dots,k-n_i^I} p_{ij}^V p_{i,k-j}^V, \forall k = 0 \dots 2n_i^I$$

2. Given $D_i^{V,m-1}$, calculate $D_i^{Vm} = D_i^{V,m-1} + D_i^V$:

$$p_{ik}^{Vm} = \Pr\{D_i^{Vm} = k\} = \sum_{j=0,\dots,k-n_i^I} p_{ij}^{V,m-1} p_{i,k-j}^V, \forall k = 0 \dots mn_i^I$$

3. Stop when $m = n_{v_i}^V$. Use the last probabilities generated, as follows:

$$p_{ik} = \Pr\{D_i = d_{ik}\} = p_{ik}^{Vn_i^V}, \forall k \in K_i$$

Remark: If n_i is large, the distribution of the above convolution may take a long time to calculate, and we may need to model those parts using other methods.

1. Decision Variables

Q_i ,	quantity ordered for part *i*
Z_{ik}^{SO},	ancillary variable for stockout of part *i*
Z_{ik}^{SS},	ancillary variable for parts *i* in safety stock that apply to the calculation of budget being used

2. Formulation

$$\min \quad \sum_{i \in I} \sum_{k \in K_i} p_{ik} c_i^{\text{SO}} Z_{ik}^{\text{SO}} \qquad (1.1)$$

subject to:

$$Z_{ik}^{\text{SO}} \text{ ³ } d_{ik} \text{ - } (q_i^0 + Q_i) \quad \text{" } i \text{ Î } I, k \text{ Î } K_i \qquad (1.2)$$

$$Z_{ik}^{\text{SO}} \geq 0 \qquad \forall i \in I, k \in K_i \qquad (1.3)$$

$$Z_{ik}^{\text{SS}} \geq q_i^0 \delta^{q^0} + Q_i - d_{ik} \quad \forall i \in I, k \in K_i \qquad (1.4)$$

$$Z_{ik}^{\text{SS}} \geq 0 \qquad \forall i \in I, k \in K_i \qquad (1.5)$$

$$\sum_{i \in I} \sum_{k \in K_i} p_{ik} c_i^{\text{SS}} Z_{ik}^{\text{SS}} \leq b \qquad (1.6)$$

$$Q_i \geq 0 \text{ and integer} \qquad \forall i \in I \qquad (1.7)$$

The above formulation prescribes order quantities for each item in order to minimize expected stockouts (or their expected cost if $c_i^{\text{SO}} \neq 1$), subject to a budget constraint on expected cost of safety stock. We can post-calculate $\sum_{i \in I} \sum_{k \in K_i} p_{ik} Z_{ik}^{\text{SS}}$ as the expected safety stock involved in the budget constraint. If $\delta^{q^0} = 1$, it will be the same as the expected safety stock. Otherwise (if $\delta^{q^0} = 0$) the actual expected safety stock will be more than the above calculation (if there was an initial stock), and can be post-calculated as $\sum_{i \in I} \sum_{k \in K_i} p_{ik} \max\{q_i^0 + Q_i - d_{ik}, 0\}$. (Note that, in the last case, we include the initial stock in the calculation of expected safety stock, but do not include it in the calculation of the cost.)
Alternative options for the objective might include minimizing expected cost of safety stock, subject to expected number (or cost) of stockouts not exceeding a given value, which can be easily formulated.

a. Formulation of Chance Constraints

Additional chance constraints, such as "the probability that a certain item has 3 or more stockouts is under 95percent can be added to the above formation. To do this, we add the following:

Additional data

$m_i^+, p_i^{\text{SO}+}$, stockout level for item *i* and maximum probability that a stockouts of that size occurs for the item. In the above example, $m_i^+ = 3, p_i^{\text{SO}+} = 1 - 0.95 = 0.05$

Additional Decision Variables

$Z_{ik}^{\text{SO}+}$, one if the *k* level of demand produces a stockout for part *i* that exceeds the maximum level, m_i^+, and zero otherwise

Additional Formulation

$$Z_{ik}^{\text{SO}+} \geq (Z_{ik}^{\text{SO}} - m_i^+)/|K_i| \qquad \forall i \in I, k \in K_i \tag{1.8}$$

$$Z_{ik}^{\text{SO}+} \in \{0,1\} \qquad \forall i \in I, k \in K_i \tag{1.9}$$

$$\sum_{k \in K_i} p_{ik} Z_{ik}^{\text{SO}+} \leq p_i^{\text{SO}+} \qquad \forall i \in I \tag{1.10}$$

Note Equation (1.8) forces $Z_{ik}^{\text{SO}+}$ to become 1 when the number of stockouts for demand level *k*, i.e., Z_{ik}^{SO}, exceeds m_i^+. Then Equation (1.10) adds up the probabilities of those levels, so as not to exceed $p_i^{\text{SO}+}$.

LIST OF REFERENCES

[1] J. Salmeron, A. Buss, T. Curling and M. Kress, “Plant utilization at Marine Corps Logistics Command,” unpublished.

[2] U.S. Marine Corps 36th Commandant’s planning guidance. 10 January 2015. J. Dunford [Online]. Available: http://www.hqmc.marines.mil/Portals/142/Docs/2015CPG_Color.pdf. Accessed March 25, 2015.

[3] Apache POI- the Java API for Microsoft documents. (n.d.). T. A. S. Foundation, [Online]. Available: http://poi.apache.org/. Accessed July 15, 2016.

[4] JFree chart. (n.d.). JFree. [Online]. Available: http://www.jfree.og/jfreechart. Accessed July 15, 2016.

[5] SimKit. (n.d.). Naval Postgraduate School. [Online]. Available: https://www.diana.nps.edu/simkit/latest/. Accessed December 14, 2014.

[6] C. Almeder, M. Preusser and R. F. Hatl, “Simulation and optimization of supply chains: Alternative or complementary approaches?,” *OR Spectrum*, vol. 31, no. 1, pp. 95–119, 2009.

[7] T. A. Lenhardt, “Evaluation of combat service support logistics concepts for supplying a USMC regimental task force,” M.S. thesis, OR Dept., Naval Postgraduate School, Monterey, CA, 2001.

[8] S. Kumar and D. A. Nottestad, “Supply chain analysis methodology—Leveraging optimization and simulation software,” *OR Insight*, vol. 26, no. 2, pp. 87–119, 2012.

[9] G. W. Godding, “A multi-modeling approach using simulation and optimization for supply-chain network systems,” Ph.D. dissertation, Dept. Systems Eng., Arizona State University, Tempe, AZ, 2008.

[10] M. Schlegel, G. Brosig, A. Eckert, M. Jung, A. Polt, M. Sonnenschein and C. Vogt, “Integration of discrete-event simulation and optimization for the design of value networks,” *Computer Aided Chemical Engineering*, vol. 21, pp. 1955–1960, 2006.

[11] F. D. Mele, G. Guillen, E. Antonio and L. Puigjaner, “A simulation-based optimization framework for parameter optimization of supply-chain networks,” *Industrial and Engineering Chemistry Research*, vol. 45, no. 9, pp. 3133–3148, 2006.

[12] Multimethod simulation software. (n.d.). AnyLogic. [Online]. Available: http://www.anylogic.com/areas/supply-chains. Accessed 22 March 2015.

[13] Top supply chain management software products. (n.d). Capterra. [Online]. Available: http://www.capterra.com/supply-chain-management-software/. Accessed 22 March 2015 .

[14] Supply chain optimization and simulation on a unified platform. (n.d.). LLamasoft. [Online]. Available: http://www.llamasoft.com/. Accessed 23 March 2015.

[15] Northrop Grumman Mission Systems, "Depot-level maintenance capacity model study," unpublished.

[16] C. T. Radsdale, "Characteristics of Optimization Problems," in *Spreadsheet Modeling and Decision* Analysis, Mason, South-Western Cengage Learning, 2012, pp. 18–20.

[17] A. Buss, "Discrete event simulation modeling," unpublished.

[18] GAMS.(n.d.). GAMS. [Online]. Available: www.gams.com. Accessed March 23, 2016.

[19] CPLEX 12. (n.d.). GAMS/CPLEX. [Online]. Available: http://www.gams.com/dd/docs/solvers/cplex/index.html. Accessed March 23, 2016.

[20] Argonne National Laboratory Toolkit for Advanced Optimization (TAO). (n.d.). U.S. Department of Energy[Online]. Available: http://www.mcs.anl.gov/research/projects/tao/. Accessed April 12, 2016.

[21] ASCEND.(n.d). Carnegie Mellon University. . [Online]. Available: http://ascend4.org/Main_Page. Accessed April 12, 2016.

[22] Gnu's Not Unix (GNU) Linear Programming Kit.(n.d.). Free Software Foundation. [Online]. Available: http://www.gnu.org/software/glpk/. Accessed April 12, 2016.

[23] NetBeans IDE 8.0.2 Information.(n.d.). Oracle Corporation. [Online]. Available: https://netbeans.org/community/releases/80/. Accessed April 23, 2016.

[24] R. Cheng, "Determining efficient simulation run lengths for real time decision making," in Winter Simulation Conference Washington D.C., United States, 2007.

[25] A. Gelman and K. Shirley, “Inference from simulations and monitoring convergence,” in *Handbook of Markov Chain Monte Carlo*, Boca Raton, Taylor and Francis Group, LLC, 2011, pp. 163–173.

[26] M. R. Chernick, *Bootstrap Methods: A Practitioner’s Guide*, New York: Wiley, 1999.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California